PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR - MATRIZ

FACULTAD DE CIENCIAS ADMINISTRATIVAS Y CONTABLES

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN ADMINISTRACIÓN DE EMPRESAS CON MENCIÓN EN GERENCIA DE LA CALIDAD Y PRODUCTIVIDAD

PROPUESTA DE MEJORA DEL PROCESO DE DESARROLLO DE SOFTWARE EN EL MÓDULO DE BANCA EN LÍNEA PARA UNA EMPRESA DE DESARROLLO DE SOFTWARE

ING. DANIEL FERNANDO JARRIN MADERA

DIRECTOR: ING. BAYARDO FLORES, MBA.

LÍNEA DE INVESTIGACIÓN: SISTEMAS DE GESTIÓN DE PRODUCCIÓN Y OPERACIONES

QUITO, MARZO 2017

DIRECTOR DEL TRABAJO DE TITULACIÓN:

MBA. BAYARDO FLORES

INFORMANTES:

MBA. PABLO LÓPEZ CHIRIBOGA

MGTR. EDWIN SUQUILLO GUIJARRO

DEDICATORIA

El presente trabajo lo dedico a Dios, por ser una fuerza inspiradora diaria para vencer mis problemas. A mi madre por su esfuerzo y dedicación diaria hacia mi persona durante todos los años de mi vida. A mi familia entera por brindarme el impulso final para lograr este objetivo.

AGRADECIMIENTO

Agradezco a Dios por brindarme la sabiduría, paciencia y fortaleza para lograr finalizar el presente trabajo de titulación.

Agradezco a mi madre por darme las herramientas y valores para poder enfrentar cualquier trabajo que se me presente en la vida.

Agradezco a mi pareja por su impulso y apoyo constante a lo largo del desarrollo del presente trabajo de titulación.

Agradezco a mis familiares y amigos por ser impulso diario en el desarrollo de cada trabajo que realizo.

Agradezco al Ing. Bayardo Flores por su asesoramiento en la dirección del presente trabajo de titulación. Y para finalizar agradezco a cada una de las personas, quienes contribuyeron para que esta etapa de mi vida concluya con éxito.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN EJECUTIVO

La mejora y correcta implementación de procesos en cualquier empresa es fundamental para mantenerse competitivos a nivel local o mundial, es así que dentro de una empresa especializada en software bancario y con un mercado extranjero muy grande, se hace primordial el ser lo más productivos posibles dentro de cada uno de sus procesos de desarrollo del software.

El presente estudio crea una propuesta de mejora al proceso de desarrollo de software del módulo de banca en línea, a través del análisis de los datos históricos de un proyecto culminado recientemente. Esta propuesta busca resolver las falencias y pérdidas de productividad en cada uno de los pasos a cumplirse a lo largo del proceso y sus dependencias. La información considerada en el análisis de la propuesta comprende el periodo abril 2016 a enero 2017. Al finalizar este estudio, se logró crear una propuesta que permite reducir el tiempo del desarrollador en un 50% al realizar una tarea mediante el uso de herramientas de eficiencia disponibles en el mercado. También, la propuesta disminuiría el tiempo de desarrollo del equipo en un 3%, al eliminar el tiempo empleado en la corrección de errores de deuda técnica, es decir se descartaría la deuda técnica del 68%. En cuanto a la versión de producto no controlada, esta propuesta permitirá reducir en un 3% el tiempo en empezar un proyecto. Además, la propuesta abarca la solución de errores de preparación de ambientes, donde el equipo de trabajo disminuiría en un 3% el tiempo total

del desarrollo del proyecto, y en un 2% adicional al eliminar los daños ocasionados por incorrecta parametrización en los ambientes de trabajo. Otro aspecto fundamental en la propuesta, es la definición de 9 nuevos indicadores que actuarán sobre el control de errores, re trabajo y productividad, que permitirán tomar acciones correctivas en el tiempo. Por último, basándonos en el uso mejores prácticas en la planificación de la metodología SCRUM, se busca cumplir en un 100% las planificaciones realizadas para las entregas parciales o sprints, respecto del actual 74%.

Este trabajo de titulación se desarrollo en 5 capítulos, el primer capítulo trata la descripción de la empresa. En el segundo capítulo, se desarrolla el marco teórico de la investigación y se tratan temas como: proceso de desarrollo de software usado en la empresa, la metodología en la que se basa el proceso y como mejorar un proceso. En el tercer capítulo, se define una línea base sobre el estado actual del proceso y sus principales falencias. En el cuarto capítulo, se detallan las soluciones a cada uno de los problemas identificados en el proceso de desarrollo de software del módulo de banca en línea, conjuntamente con los indicadores que nos permiten controlar al mismo. Finalmente, en el quinto capítulo se desarrollan las conclusiones y recomendaciones del análisis realizado.

INTRODUCCIÓN

La empresa de software bancario sobre la cual hemos basado este trabajo de titulación, busca activamente una mejora continua de sus procesos. Dentro de la empresa, existen varios módulos especializados sobre pequeñas partes de un core bancario y cada uno de ellos cuenta con equipos de trabajo. Uno de los módulos mencionados es el módulo de Banca en Línea, y del cual se ha visto podría ser mejorado en cuanto a procesos ya que se han detectado ciertas falencias, principalmente en la parte de planificación, así como también dependencias con otros módulos que causan grandes retrasos.

La gerencia de proyectos busca constantemente se cumpla la planificación realizada con el equipo de trabajo usando la metodología actual de desarrollo de software denominada SCRUM. Esta metodología se basa en los principios de inspección continua, adaptación, auto-gestión e innovación, y su forma de trabajo busca entregas cortas y manejables denominadas sprints. Además, la metodología SCRUM se complementa con el marco de trabajo SAFe para poder escalar las metodologías ágiles a toda la empresa. Básicamente esta metodología y marco de trabajo se complementan perfectamente con el uso de procesos para la solución de tareas en un tiempo definido, porque poseen un procedimiento claro a seguir.

La información considerada en el análisis de la propuesta comprende los históricos de un proyecto específico dentro del período abril 2016 a enero 2017. Este proyecto sirvió de

base para encontrar las principales falencias en el proceso de desarrollo de software del módulo. Para el análisis de los datos nos basaremos en la metodología del caso único, debido a que los proyectos dentro del área suelen durar entre semestres o años y no se tienen más datos históricos de otros proyectos que compartan la misma problemática del módulo de canales. Dentro de cada área y módulo, a pesar de compartir el mismo proceso de desarrollo de software, tienen sus propios problemas debido a la tecnología y dependencias propias con otros módulos. La metodología de caso único tiene las ventajas de proveer la capacidad de establecer las condiciones y naturaleza de una relación entre causa-efecto, la capacidad de trabajar en situaciones atípicas por motivos como los antes descritos, y por último nos provee un valor persuasivo, pudiendo convertir en ideas concretas, conceptos considerados como principios abstractos. En cambio, las desventajas de este método de caso único son el no poder generalizar sus propuestas o conclusiones, porque estadísticamente un muestreo mayor provee más confiabilidad en las mismas, y la variabilidad por diversos factores de los proyectos dentro de la empresa de software pueden ocasionar cierto error en las inferencias del estudio.

Los objetivos de este trabajo, comprenderán:

- Elaborar una propuesta con mejoras al proceso de desarrollo de software del módulo de Banca en Línea, enfocándonos en tener una mayor productividad y control sobre el mismo.

- Identificar las falencias y pérdidas del actual proceso de desarrollo de software del módulo de Banca en Línea.

- Definir indicadores del proceso de desarrollo de software para el módulo de Banca en Línea, que me permitan tener un mejor control sobre el mismo.

- Definir un esquema de mejora continua para el proceso de desarrollo de software en el módulo de Banca en Línea.

Este documento está conformado por cinco capítulos, que comprende en una primera etapa la descripción de la empresa. En un segundo capítulo, se trata el desarrollo teórico de los temas que conforman el proceso de desarrollo de software de la empresa, como por ejemplo la metodología en la que se basa el proceso y sus principales componentes. En el tercer capítulo, se identifica una línea base del estado actual del proceso, definiendo paso a paso el flujo del mismo, conjuntamente con sus entradas, salidas, dependencias, y límites. Además en esta tercera parte se identifican las principales falencias que tiene el proceso en cuanto a productividad y control sobre el mismo mediante el uso de diagramas de causa y efecto. En el cuarto capítulo, se detallan las soluciones a cada uno de los problemas encontrados en el proceso de desarrollo de software del módulo de banca en línea, conjuntamente con los indicadores que nos permitan tener un mejor control sobre el mismo y a su vez tomar acciones correctivas en un futuro. Además, se define un esquema de mejora continua basada en las reuniones de retrospectiva de la metodología SCRUM. Para culminar, se encontrará en el quinto capítulo las conclusiones y recomendaciones del análisis realizado.

Desde el punto de vista aplicativo el estudio aporta con herramientas, mejores prácticas y mejoras a los procesos, al momento de desarrollar software con metodologías ágiles como SCRUM y tomando en cuenta el marco de referencia denominado Scaled Agile Framework SAFe. Además, provee un esquema de mejora continua basado en

metodologías ágiles que puede ser aplicado a nivel institucional. En el ámbito estratégico institucional, el estudio aporta con indicadores que ayudarán a tomar decisiones a niveles gerenciales. Estas decisiones tendrán un fundamento cuantificable que permita justificarlas en el tiempo.

1. ANTECEDENTES GENERALES Y DIAGNÓSTICO SITUACIONAL

1.1 DATOS DE LA ORGANIZACIÓN O INSTITUCIÓN

El bajo costo de la mano de obra en el Ecuador con respecto a países europeos y norteamericanos, conjuntamente con una alta demanda de programadores de software a nivel mundial, han permitido que la industria del software ecuatoriana crezca los últimos 20 años, y sea considerada como una industria estratégica por el gobierno. De acuerdo con (Pascual, 2015), un programador norteamericano gana en promedio $83.000 dólares anuales, un programador europeo $53.000 dólares anuales, en comparación a un ecuatoriano que gana en promedio $20.000 dólares anuales. Hay otros países latinoamericanos que también tienen sueldos bajos en comparación con países europeos o norteamericanos como Perú y Colombia con $15000 dólares anuales, o Argentina con $31000 dólares anuales. Sin embargo, los programadores ecuatorianos han ganado una excelente reputación debido a que se exporta en promedio 50 millones de dólares anuales en software de muy buena calidad según (Rodriguez, 2016). Estas cifras de exportación son el resultado de estadísticas y encuestas extrapoladas a toda la industria, ya que no existen datos oficiales. Cabe recalcar, que esta cifra puede aumentar debido a que la mayoría de programas no pasan por la aduana y no siempre se registra la exportación, y lo que es más común, es que empresas que crean el software lo venden a empresas locales y estas a su vez

lo comercializan a una empresa extranjera, lo cual dificulta aún más el registro de exportación. Además, los bancos y las empresas financieras se decantan por usar mano de obra ecuatoriana en desarrollo de software, porque existen varias empresas ecuatorianas que ofrecen una solución o software integral que cubre varias necesidades o funcionalidad del mercado, y donde ya se encuentra inmerso el costo de mano de obra en el producto final, en comparación con software desarrollado en otros países, el costo está por debajo.

Una de las empresas que conforman el crecimiento antes nombrado, es la empresa analizada. Dicha empresa busca ser competitiva a nivel internacional, razón por la cual ha implementado nuevas metodologías de desarrollo de software como SCRUM y el marco de trabajo SAFe, que le permitan vender sus productos y servicios software de manera diferente, y al mismo tiempo desarrollar e implementar los mismos a través de entregas cortas y manejables denominadas sprints. Para poder cumplir las entregas cortas, la empresa debe tener mejor control sobre cada uno de los procesos de desarrollo de software, en especial el módulo de Banca en Línea, debido a que es uno de los productos prometedores. Es importante resaltar que muchos de los datos de la empresa no han sido revelados por motivos de políticas de seguridad.

1.2 ÁREAS Y TAMAÑO DE LA EMPRESA

La empresa consta en su nómina de aproximadamente 100 desarrolladores en el área de ingeniería y desarrollo de nuevos productos de software, 25 de esos

desarrolladores conforman el área de canales. El área de canales se encarga del desarrollo de los módulos de Banca en Línea, Cajeros Automáticos y Banca Móvil. Generalmente, se encuentran asignadas 14 personas al módulo de Banca en Línea, entre los que están: 2 equipos de desarrolladores de 4 personas, 2 System Architects, 1 Product Manager, 1 Product Owner y 2 Scrum Master. En el capítulo del marco teórico se especificarán las funciones de los roles antes nombradas de acuerdo al Scaled Agile Framework. Además, se debe aclarar que el módulo de Banca en Línea consta de dos submódulos: Banca en Línea Personas, y Banca en Línea Empresarial. Regularmente cada submódulo es atendido por uno de los equipos de desarrollo y son guiados o seguidos por el Product Manager, Product Owner, System Architect, y Scrum Master en base a SCRUM y el marco de trabajo Scaled Agile Framework SAFe. Por otro lado, las áreas que maneja y complementan a la empresa son las de pasivas, activas, y generales. El área de pasivas consta de los módulos de cuentas corrientes, cuentas de ahorros, cámara, remesas, servicios bancarios, compra-venta de divisas, emisión de cheques de gerencia y comercio exterior. El área de activas por su parte está conformada por los módulos de cartera, créditos, garantías, originación, cobranzas y tesorería. Por último, el área de generales abarca los módulos de clientes, administración, seguridades y parametrización. Todas las áreas de la empresa se relacionan con el área de canales, y específicamente el módulo de Banca en Línea, debido a que el mismo usa muchas funcionalidades ya definidas en las áreas de pasivas, activas y generales. Varias veces el proceso de desarrollo de software se resume y limita en generar la capa intermedia que conecta desde el sitio web a la funcionalidad de las otras áreas y módulos de la empresa, mediante el uso de servicios web. Normalmente, el volumen del software que produce el módulo de Banca en Línea es bastante variable y muchas veces se traduce en aumentar

funcionalidad cuando el cliente ya previamente tiene una banca en línea o incorporar toda la banca en línea que tiene el módulo. Si el proyecto trata sobre aumentar funcionalidad se deben analizar temas como convivencia tecnológica y migración de información. Cuando se tratar de incorporar una funcionalidad completa se minimiza el trabajo solo a la migración de información. Por las razones antes nombradas, los desarrollos de software se vuelven tan complejos como el cliente lo pida y tenga información y no son tareas repetibles en un tiempo estándar.

1.3 PROCESOS, METODOLOGÍAS Y MARCOS DE TRABAJO DE LA EMPRESA

Actualmente la empresa maneja un solo proceso de desarrollo ágil de desarrollo de software para todas las áreas y módulos de la empresa. Este proceso ágil de desarrollo de software se basa en la metodología SCRUM y el marco de trabajo SAFe. La metodología de SCRUM es implementada a nivel de equipos de trabajo en toda la organización, pero al mismo tiempo es escalada a toda la empresa a través del marco de referencia denominado Scaled Agile Framework o lo que se conoce por sus siglas en inglés SAFe. Este marco de referencia está basado en SCRUM y principios ágiles, para ayudar a ganar agilidad a la empresa a nivel de organización y ya no solo de equipo.

El principal objetivo de SAFe para una empresa en general, es incorporar un sistema con prácticas probadas e integradas con el fin de llevar mejoras sustanciales en el compromiso de los empleados, calidad de las soluciones, tiempo de comercialización y productividad dentro de los equipos de trabajo. Estas mejoras permitirán dar un giro completo a nivel cultural empresarial para que la empresa siga siendo una de las empresas exportadoras de software del país.

Cabe recalcar, que tanto la metodología SCRUM como el marco de referencia SAFe se complementan muy bien con los procesos, debido a que las dos promueven un manejo de procesos interno con ciertos insumos y salidas, con el fin de buscar soluciones de calidad y ágiles en toda la empresa. Dentro del proceso de desarrollo de software ágil que se aplica para toda la empresa, se han detectado ciertas falencias en la aplicación del mismo en el módulo de Banca en Línea, principalmente en la parte de planificación, así como también dependencias con otros módulos que causan grandes retrasos.

Además de las falencias antes nombradas en el proceso, se han identificado oportunidades de mejora y herramientas que pueden ser usadas con el objetivo de mejorar la productividad y aligerar la carga de trabajo para los equipos ágiles. Estas herramientas de trabajo están siendo usadas por empresas pioneras del desarrollo de software y permiten disminuir los costos notablemente del producto.

Por otro lado, la empresa busca un esquema de mejora que permita corregir continuamente cada uno de los procesos de desarrollo de software de la empresa en el tiempo.

1.4 OBJETIVOS E INDICADORES DEL ÁREA DE CANALES

La gerencia de la empresa de acuerdo a su visión estratégica, año tras año transmite los objetivos que deben cumplir todas las áreas de la empresa, entre ellas el área de canales. Es decir, cada área maneja sus propios objetivos medibles a través de indicadores y son alineados con la visión estratégica de la empresa a través de la

gerencia. Estos objetivos son definidos a partir de resultados obtenidos en el año anterior y buscan la mejora continua. Los principales indicadores que maneja la empresa para las áreas son:

Tabla 1:
Indicadores de la empresa

Tipo	Categoría	Nombre	Fórmula
Indicadores Actuales	Metodología SCRUM	Velocidad del Equipo	Número de puntos de historia finalizados en un Sprint
		% Cumplimiento por Sprint	(Número puntos de historia aceptados /Número de puntos de historia comprometidos)* 100
		Valor del Negocio Entregado al Cliente	Valor total de negocio asociado con las funcionalidades entregadas en un Sprint
		% Deuda Técnica	(Valor de deuda técnica de Sprint expresada en puntos / Valor de puntos de historia realizados en Sprint)*100
	Productividad	Costo unitario Producción	Costo de célula por Sprint / Suma puntos de Historia realizados en Sprint
		Productividad por Sprint	((Número de funcionalidades expresadas en Puntos de Historia entregados por sprint * 8)/ Número de horas trabajadas equipo)
	Efectividad	% de Cumplimiento de Trabajo objetivo Sprint	(Número de horas ejecutadas Sprint/ Número de horas objetivo planificadas Sprint)*100
	Eficiencia	Eficiencia Por Sprint	((Número de funcionalidades de sprint expresadas en Puntos de Historia planificados * 8)/ Número de horas trabajadas equipo)
		Costo por Horas Hombre en Sprint	Costo Total Célula en Sprint / Total Horas Ejecutadas
	Errores	Errores por sprint	Número de Errores encontrados en el sprint

Estos indicadores sirven para identificar el estado del proceso de desarrollo de software que es compartido por toda la empresa y que nos permita definir objetivos de mejora en base a ellos. Varios de estos indicadores hablan sobre puntos de historia. Los puntos de historia son propios de la metodología SCRUM y se refieren a la complejidad que tiene una historia para realizarse, sin embargo para poder cuantificar métricas en base a horas de trabajo a cada punto de historia se le toma

como un día de trabajo o 8 horas hombre. Para asignar los valores de puntos de historia que tiene una historia, normalmente el Product Owner se basa en la serie de fibonnaci hasta el número 8. Los valores válidos de una historia de trabajo serán: 1,2,3,5,8. Si una historia necesita un mayor valor de debe dividir en 2 porque de lo contrario no es efectivo el seguimiento de la misma.

Los objetivos que se definieron por parte de la gerencia para el área de canales en el año 2017 son:

- La Velocidad de un Equipo Ágil de desarrollo debe mejorar en 15% durante el año.

- El cumplimiento del equipo ágil del sprint debe ser mínimo del 95%.

- El valor del negocio entregado al cliente cada sprint por equipo ágil debe ser en promedio de 30 puntos.

- El promedio de deuda técnica por equipo ágil debe reducirse en 50%.

- El costo unitario de producción debe reducirse en un 15% para las células o equipo ágiles.

- Se debe mejorar en un 15% la productividad en sprint para las células o equipos ágiles.

- El costo por horas hombre en sprint debe disminuir en un 15%.

- La eficiencia del equipo ágil por sprint debe aumentar en un 15%.

- El promedio de errores por sprint debe disminuir en 50%.

1.5 VISIÓN DEL PRODUCTO BANCA EN LÍNEA DENTRO DEL ÁREA CANALES

La empresa ha apostado por el módulo de Banca en Línea y su desarrollo porque es una especie de punta de lanza para poder penetrar nuevos mercados. Estos nuevos mercados se han desarrollado gracias a la tendencia de las nuevas generaciones de personas al hacer los trámites bancarios usando el internet y evitarse ir a entidades bancarias como era el procedimiento normal. En la actualidad, el ahorro de tiempo para dedicarlo a otras actividades es muy valioso. Para poder penetrar a la mayor cantidad de mercados, el producto que nace desde el módulo de banca en línea debe concebir funcionalidad de la manera más genérica posible para que no se encuentre atado a leyes de países específicos y sea fácil su implementación en varias entidades financieras o bancos. Normalmente la persona que se encarga de tener esta visión clara y trasmitir a todo el equipo es el Product Manager. El rol del Product Manager es un rol de Scaled Agile Framework y tiene obligaciones y derechos propios con el marco de trabajo. A parte de estas obligaciones y derechos, el Product Manager debe tener en mente el mercado, los productos ofrecidos por las empresas competencia, y si el producto empata con el modelo de negocio de la empresa. Es decir, el Product Manager debe tener una comunicación muy clara con el área de ventas y gerencial de

la empresa para nos desviarse de lo que quiere la empresa. En este caso, la visión de la empresa sobre el producto es que sus funcionalidades deben ser lo más genéricas posibles, para poder implementarse en distintos países y las páginas web deben poder mostrarse tanto en dispositivos móviles como navegadores de computadores normales. La característica antes nombrada permitirá que el producto sea usado por una mayor cantidad de usuarios y nos obliga usar tecnologías como HTML5 y CCS3.

2. MARCO TEÓRICO

2.1 LA INDUSTRIA DE SOFTWARE EN EL ECUADOR

De acuerdo con (El Comercio, 2014) existen alrededor de 600 empresas dedicadas al desarrollo de software en el Ecuador. La gran mayoría de estas empresas son medianas y pequeñas empresas que se dedican al desarrollo de soluciones para empresas de tamaños similares. La mayoría de sistemas desarrollados por las compañías mencionadas son CRM's (administración de relación con los clientes), ERP's (planificación de recursos empresariales), ERM's (gestión de riesgos), y programas para la gestión de logística y aplicaciones móviles. Además se estima que en el año 2015, estas 600 empresas vendieron alrededor de 550 millones de dólares.

Cabe indicar que solo 144 empresas son registradas en la Asociación Ecuatoriana de Software y se encuentran ubicadas 111 en Quito, 20 en Guayaquil, 3 en Cuenca, 3 en Ambato, 2 en Latacunga, 1 en Lago Agrio, 1 en Loja y 3 en Manta. De estas 144 empresas existen 10 empresas que se dedican a la exportación del software, y 5 de ellas se dedican a atender al sector financiero de países como Perú, Honduras, Guatemala, Costa Rica, México, Colombia, Venezuela, Chile, Argentina y Croacia. El software bancario ecuatoriano ha ganado mercado por ser menos costoso que el procedente principalmente de Estados Unidos, Unión Europea y Asia (AESOFT, 2016).

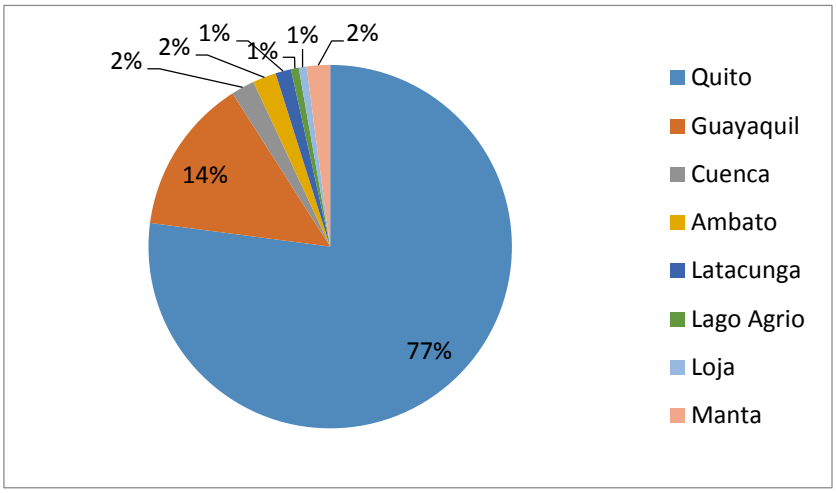

Figura 1 Empresas dedicadas al Desarrollo de Software por Provincia en Ecuador
Fuente. AESOFT. (7 de Enero de 2016). *AESOFT*. Recuperado el 17 de Enero de 2016, de http://aesoft.com.ec/?page_id=3

Según (AESOFT, 2016), la importancia de las tecnologías de la información y en específico del software, es que su accionar es transversal a todos los sectores. En consecuencia, el Software en el Ecuador fomenta lo siguiente:

- Crea grandes fuentes de empleo.
- Competitividad para todos los sectores.
- Se adapta a los requerimientos del mercado y origina soluciones de valor.
- Crea valor e impacto con nivel de ganancias tempranas.
- Mejora capacidad de atraer inversiones.

- Usa talento humano de alto valor agregado.

Con el objetivo de desarrollar la industria de software en el Ecuador, alineada con el cambio de la matriz productiva, la AESOFT ha definido una estrategia con 6 puntos claves como son: Marco legal, Calidad y Certificación, Promoción e Internacionalización, Acceso a Financiamiento, Capacitación, y Ambiente de Negocios.

Figura 2 Ejes de Acción Estrategia AESOFT

Fuente. AESOFT. (7 de Enero de 2016). *AESOFT*. Recuperado el 17 de Marzo de 2016, de http://aesoft.com.ec/?page_id=38

Tomando en cuenta la decisión estratégica que ha realizado la principal asociación que tiene el país, nos enfocaremos en el punto de calidad y certificación, donde podemos visualizar que para que una empresa pueda adaptarse a las necesidades actuales del mercado ecuatoriano, debe orientar a todos sus procesos de desarrollo de software hacia la calidad y certificación de la misma. Al hacer uso de metodologías

ágiles y la certificación en las mismas, las empresas buscan cumplir el nivel de calidad exigido en los mercados.

2.2 METODOLOGÍAS DISPONIBLES PARA EL MANEJO DE PROYECTOS DE SOFTWARE

2.2.1 Generalidades

Muchas veces las metodologías en el contexto de ingeniería de software son un concepto difícil de entender, pero después de varios años de usarlas, a mi opinión una metodología de desarrollo de software es un criterio común entre un equipo de trabajo para realizar las tareas del software deseado. Dicho esto y debido a que el campo de desarrollo de software es altamente cambiante por los lenguajes de programación usados y las plataformas sobre las cuales corren los mismos, se han propuesto varias metodologías a lo largo de los años.

2.2.2 ¿Qué metodologías debo usar?

Inicialmente teníamos metodologías desarrolladas en los años 90 que denominábamos tradicionales o pesadas, y buscaban el proveer pautas para realizar cada una de las tareas del desarrollo del software. Básicamente pretendían hacer a un proyecto informático como una extensión de un proyecto burocrático tradicional, desencadenando en productos que no toman en cuenta la calidad, satisfacción del cliente, y competitividad.

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar (Acuña, 2009).

Las principales metodologías tradicionales son:

- **RUP (Rational Unified Procces):** Es una metodología basada en un proceso continuo de pruebas y retroalimentación, que permite abarcar grandes escalas. El principal problema de esta metodología es el poder crear controles de administración sobre el mismo.

- **MSF (Microsoft Solution Framework):** Esta metodología se basa en modelos de proceso y relega a segundo lugar al aspecto tecnológico de los proyectos. Principalmente busca la relación de modelos y mejores prácticas que se encargan de la gestión, planificación, y desarrollo de proyectos de software.

- **Iconix:** Es una metodología basada en un conjunto de métodos orientados a objetos que limita su acción a todo el ciclo de vida de un proyecto. Esta metodología baso su origen, en la practicidad de desarrollo que tiene la metodología XP (Extreme Programming) y en la complejidad de análisis de la metodología RUP (Rational Unified Processes).

La mayoría de los métodos antes nombrados quedaron muy obsoletos en este mundo cambiante y más tarde aparecieron las metodologías que denominamos ágiles, las cuales buscan hacer frente a los cambiantes requisitos de los clientes y al mismo tiempo que sean rápidas en su desarrollo. Principalmente lo que se quiere es tener la capacidad de adaptación a los cambios de manera ágil.

Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas pragmáticas, que harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega (Acuña, 2009).

Las principales metodología ágiles del mercado son:

- **ASD (Adaptive Software Development)**: Esta metolología tiene como eje central la adaptación continua a circunstancias cambiantes, debido a que no busca el desarrollo mediante el típico ciclo de desarrollo de software: planificación, diseño, y construcción, sino más bien con un ingenioso

ciclo: especular, colaborar y aprender. Dentro de la metodología se reconoce que su funcionamiento es cíclico y que en cada iteración se producirán cambios e inclusive errores.

- **XP (Extreme Programming):** Esta metodología es muy útil cuando se tienen requisitos cambiantes e imprecisos y donde existe un alto riesgo técnico, debido a que se basa en potenciar las relaciones interpersonales del equipo de desarrollo de software, mediante el trabajo en equipo, procurando el aprendizaje de los desarrolladores, y buscando un buen clima laboral.

- **DSDM (Dynamic Systems Development Method):** Esta metodología se basa en la la continua interrelación del usuario con el desarrollo creciente e iterativo, que sea sensible a los requerimientos cambiantes, buscando cumplir con los requerimientos financieros y de tiempo del proyecto.

- **SCRUM:** Esta metodología se basa en los principios de inspección continua, adaptación, auto-gestión e innovación, y su forma de trabajo es hacer entregas cortas y manejables denominadas Sprints. Esta metodología permite disminuir riesgos al hacer un proyecto de forma colaborativa. También, es ideal para proyectos que se desarrollan en entornos cambiantes y complejos que necesitan rapidez en resultados y flexibilidad al mismo tiempo. Su origen se da en el año de 1986, cuando se basan en procesos exitosos de empresas estadounidenses y japonesas como Xerox, HP, y Honda. Estos procesos buscaban producir productos que iniciaban a

partir de requisitos novedosos y generales, pero que necesitaban ser sacados al mercado de manera mucho más ágil. En el año de 1995 esta metodología fue formalizada. Actualmente, la empresa sobre la cual se está buscando la mejora de proceso, usa esta metodología, y la analizaremos más a fondo en los siguientes capítulos.

2.2.3 Metodología SCRUM para gestión de proyectos de software

Como mencionamos anteriormente el proceso de desarrollo de software en la empresa de estudio se basa en SCRUM conjuntamente con Scaled Agile Framework. La metodología SCRUM tuvo su origen a partir de un estudio sobre procesos exitosos de empresas japonesas y estadounidenses como Xerox, Hp y Honda en el año de 1986. Estos procesos eran enfocados en producir productos que partían de requisitos novedosos y generales, pero que necesitaban ser entregados al mercado de manera mucho más rápida. En estas empresas se identificaron patrones bastante parecidos entre ellas al momento de la ejecución del proceso. Es así, que al identificar estos patrones en el estudio antes nombrado, se logró comparar la forma de trabajar de estos equipos muy productivos y colaborativos con la formación de Rugby SCRUM.

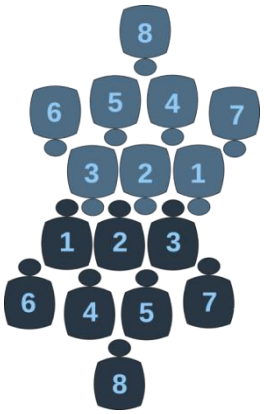

Figura 3 Formación de Rugby SCRUM
Fuente. World Rugby. (7 de Enero de 2009). *Leyes del Juego de Rugby*. Recuperado el 10 de Junio de 2016, de http://laws.worldrugby.org/?variation=1&law=20&language=ES

Desde el año de 1995 la metodología de SCRUM fue formalizada, y desde ese año varias empresas multinacionales alrededor del mundo las usan en cada uno de sus proyectos.

Tabla 2:
Empresas Multinacionales que han usado metodología SCRUM

Sectores	Empresas
Banca	Barclays Global Investors Merrill Lynch Bank of America
Internet	Amazon Yahoo Google
Media y Telecomunicaciones	BellSouth Motorola Nokia Motorola Sony/Ericsson Verizon Telefónica I+D
Software y Hardware	Adobe Citrix IBM Intel Microsoft

Esta metodología básicamente prioriza el desarrollar la funcionalidad de mayor valor para el cliente, e involucra directamente al mismo para que poco a poco se entusiasme y se compromete con el proyecto, debido a que lo ve crecer iteración a iteración.

La manera en que se trabaja en SCRUM es mediante entregas parciales y regulares denominadas sprints, y que previamente han sido priorizadas por el beneficio que aportan al cliente y empresa. La principal ventaja de esta metodología se da en entornos complejos, donde se ha identificado la necesidad de obtener resultados rápidamente, y cuando los requisitos son muy variables, conjuntamente con la necesidad de una alta flexibilidad y productividad.

Los participantes de una metodología SCRUM son:

- **Product Owner o dueño del producto:** Es el encargado de hablar por el cliente, y busca que el equipo cumpla las expectativas funcionales en cada característica del producto que el mismo defina. Sus principales funciones se resumen en:

1. Definir correctas historias de usuario.

2. Decidir qué características de producto construir y cuáles no.

3. Generar criterios de aceptación para las nuevas funcionalidades.

4. Definir productos mínimos viables para el mercado

5. Priorizar las historias de usuario en base a criterios de valor que la empresa le entregue.

- **Scrum Master:** Principalmente es el encargado de minimizar los problemas para cumplir el objetivo del Sprint, y es la persona que lidera las reuniones de pie. Sus funciones se resumen en:

1. Soluciona impedimentos o problemas que se dan al momento de desarrollar la iteración el equipo ágil.

2. Enseña al Product Owner a administrar y priorizar el product backlog del proyecto.

3. Se encarga de que todos los miembros del equipo ágil entiendan la definición de una tarea realizada o done.

4. Junto al equipo actualiza el progreso de su trabajo para alimentar a las métricas diarias.

5. Busca y promueve las buenas prácticas de programación.

6. Se coordina con el product owner para maximizar el valor del negocio.

7. Lidera las reuniones de pie y de retrospectivas.

- **Agile Team o equipo ágil:** Básicamente son el equipo de desarrolladores de software comprometidos a cumplir lo planificado en el sprint. Sus principales funciones se resumen en:

1. Transformar el backlog del producto en software completamente funcional.

2. Tener una mentalidad de trabajo en equipo.

3. Transmitir el conocimiento funcional y técnico a cada uno de los integrantes del equipo.

4. Tener apertura a la capacitación y buscar ser un equipo multidisciplinario.

5. Cumplir el compromiso planificado en la reunión de planificación.

- **Cliente:** Es quien recibe el producto y puede aportar con ideas para mejoras al producto.

Por otro lado, el procedimiento a seguir que nos permitirá trabajar en sprints es el siguiente:

- **Product Backlog:** Es una lista de deseables sobre las funcionalidades del producto. Es elaborado por el Product Owner y las funciones están priorizadas según lo que es más y menos importante para el negocio. El objetivo es que el Product Owner responda la pregunta "¿Qué hay que hacer?" (Gutiérrez, 2014). A continuación detallaremos las partes de las que se compone un product backlog:

1. Un backlog está compuesta por los requisitos de alto nivel del producto, y generalmente son expresadas en historias de usuario con sus respectivos criterios de aceptación. Para cada uno de los requisitos se debe indicar el valor que aporta al cliente y el costo estimado para realizarlo. Esta lista debe estar balanceada y priorizada entre el coste estimado que tiene la solución de cada requisito, con el valor que representa para la empresa. Lo que se busca es maximizar el retorno sobre la inversión ROI.

2. La lista consta de las posibles iteraciones que espera el cliente para que se le entreguen su requerimiento. Las posibles iteraciones deben ser calculadas en base a la velocidad promedio del equipo.

3. Esta lista debe incluir posibles riesgos que se puedan dar alrededor de la entrega de los requerimientos, conjuntamente con las tareas para mitigar dichos riesgos.

- **Sprint Backlog:** Es un subconjunto de ítems del Product Backlog, que son seleccionados por el equipo para realizar durante el Sprint sobre el que se va a trabajar. El equipo establece la duración de cada Sprint (Gutiérrez, 2014). Generalmente los sprints son definidos en un lapso de tiempo de 2 semanas y se debe considerar la velocidad del equipo para añadir historias de usuario a ser resueltas en la misma.

- **Sprint Planning Meeting:** Esta reunión se hace al comienzo de cada Sprint y en ella se define cómo se va a enfocar el proyecto que viene del Product Backlog las etapas y los plazos. Cada Sprint está compuesto por diferentes características o features. Por ejemplo, decidimos que los features del primer Sprint son: diseño del logo, definición colores y contenido multimedia (Gutiérrez, 2014). Esta reunión se compone de los siguientes pasos:

1. El Product Owner o dueño del producto comenta el objetivo del sprint y resume el product backlog.

2. El equipo asigna estimaciones de tiempo y divide a las historias de usuario en las tareas que sean necesarias. En caso de existir dudas, el encargado en aclarar las mismas será el Product Owner y los miembros del equipo ágil.

3. El equipo selecciona las historias de usuario que van a ser incluidas en la iteración o sprint. En este pase se debe verificar si la velocidad del

equipo permitirá cumplir con las historias propuestas. Una vez se ingresan las historias al sprint, el equipo se compromete a cumplirlas en el período de la iteración.

4. Se selecciona una hora y un lugar para la reunión de seguimiento diaria de la iteración o sprint. Además, se define la hora, el día, y el lugar para la presentación de una Demo de lo desarrollado.

- **Daily Meeting o Stand-up Meeting:** Es una reunión breve que se realiza a diario mientras dura el período de iteración o sprint. Se responden individualmente tres preguntas: ¿Qué hice ayer?, ¿Qué voy a hacer hoy?, ¿Qué ayuda necesito? El Scrum Master debe tratar de solucionar los problemas u obstáculos que se presenten (Gutiérrez, 2014). Es importante resaltar que una correcta Daily meeting o reunión diaria debe ayudar a empezar correctamente el día, nos debe ayudar a comunicar qué está pasando con el trabajo realizado en la iteración o sprint, nos debe ayudar a resolver problemas, y por último nos debe ayudar a reforzar el sentido de equipo en cada uno de los integrantes. Esta reunión se compone de los siguientes pasos:

1. Se reúnen a la hora planificada el Scrum Master y los miembros del equipo ágil.

2. Cada uno de los miembros del equipo ágil responde las preguntas ¿Qué hice ayer?, ¿Qué voy a hacer hoy?, ¿Qué ayuda necesito?, ¿Qué impedimentos o problemas respecto a la tarea tengo?.

3. El Scrum Master da por cerrada la reunión y ayuda a eliminar los impedimentos encontrados en la reunión. Estos impedimentos pueden involucrar a otras áreas de la empresa, así que él es el encargado de gestionar la comunicación hacia ellas.

- **Sprint Review:** Se revisa el sprint terminado, y ya debería haber un avance claro y tangible para presentárselo al cliente (Gutiérrez, 2014). Generalmente un Sprint Review no dura más allá de 4 horas y consta de los siguientes pasos:

1. El equipo procede a realizar una demo al product owner y los skateholders de las funcionalidades completadas en el día, lugar y fecha acordada.

2. La funcionalidad deberá ser presentada desde un servidor lo más parecido a producción, y la misma empieza a ser presentada por cualquier miembro del equipo ágil.

3. Otro miembro del equipo ágil comienza presentando las metas del sprint, el product backlog comprometido, y el product backlog completado. Mientras se realiza la presentación se pueden ir solventando dudas y descubriendo si los skateholders desearían algún cambio a nivel de la funcionalidad, para ponerlo a consideración. También, los skateholders pueden identificar si alguna funcionalidad no ha sido entregada

correctamente o pueden poner a consideración el agregar una nueva para complementar la actual.

4. Al final del sprint review, el Scrum Master deberá informar al Product Owner y skateholders, donde se realizará la próxima reunión de sprint review.

- **Sprint Retrospective o reunión de retrospectiva:** En la reunión de retrospectiva el equipo revisa los objetivos cumplidos del Sprint terminado. Se identifica lo bueno y lo malo, para no volver a repetir los errores. Esta etapa sirve para implementar mejoras desde el punto de vista del proceso del desarrollo (Gutiérrez, 2014). Generalmente la duración de una reunión de retrospectiva son 2 horas y sus principales fases son:

1. El Scrum Master explica el objetivo de la reunión retrospectiva y describe los diferentes pasos y duración que va a tener la misma.

2. El Scrum Master da a conocer los objetivos cumplidos, y los no cumplidos. Para esto el Scrum Master hará uso de las métricas o indicadores que nos proporciona la metodología SCRUM. Los miembros del equipo proceden a identificar los problemas que se tuvieron para el no cumplimiento de los objetivos planificados. Regularmente los equipos ágiles usan la herramienta de lluvia de ideas para realizar la identificación de los problemas. Luego, se procede a asignar un valor por

votación para definir qué problemas afectaron en mayor cantidad. Para estas votaciones el Scrum Master proveerá el material necesario.

3. Para los problemas que han afectado en mayor cantidad a la iteración o sprint se identifican las principales causas. Para identificar las casusas antes nombradas se puede hacer uso del diagrama de espina de pescado o causa efecto. Es importante concientizar, que en ningún momento se deben buscar culpables, sino más bien identificar los puntos a mejorar que tiene el proceso de desarrollo de software, la interacción de equipo o la comunicación con terceros.

4. Se define un plan de acción en base a soluciones rápidas y viables en los tiempos cortos de entrega. Estas soluciones pueden nacer de una lluvia de ideas por parte de todo el equipo, y más tarde mediante votación llegar a un consenso para escoger una. Los criterios para evaluar una solución deben ser esfuerzo, costo, y tiempo que se empleará.

5. Con las soluciones aprobadas, el equipo procede a descomponerlas en tareas para que puedan ser puestas en el product backlog o directamente en la lista de tareas de la siguiente iteración o sprint.

6. En la reunión de retrospectiva también se deben reconocer las cosas buenas que se hicieron, de esta forma se podrá potenciarlas y seguirlas repitiendo en próximas iteraciones o sprints.

7. El Scrum Master cierra la reunión, una vez revisado lo bueno y lo malo que se ha realizado en el sprint, y se ha definido las acciones con el esfuerzo que involucra resolverlas.

El esquema que resume el flujo, los componentes y los roles de la metodología SCRUM, se detalla en la figura de la parte inferior.

Figura 4 Metodología SCRUM
Fuente. Gutiérrez, C. (2014). ¿Qué es SCRUM?. Recuperado de: http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metogologia-agil

Dentro de la metodología SCRUM manejaremos ciertos indicadores que son estándares y propios de la misma. Según (PMOInformatica.com, 2012), estos indicadores son:

- **Valor Puntos de Tarea en Desarrollo**: Este indicador nos muestra de cierta manera la productividad del equipo, y se basa en el principio que dice "a menor trabajo en proceso, existe una mayor productividad". La

razón por la cual este principio expresa lo antes nombrado, es porque da a suponer que las historias que están siendo procesadas, se están resolviendo de forma óptima.

Valor Puntos Tareas en Desarrollo = Suma del valor de puntos de historia que se están desarrollando en el momento.

Se pueden realizar comparaciones en el tiempo para ver como el equipo ha ido resolviendo historias, y dónde han tenido la mayor cantidad de inconvenientes.

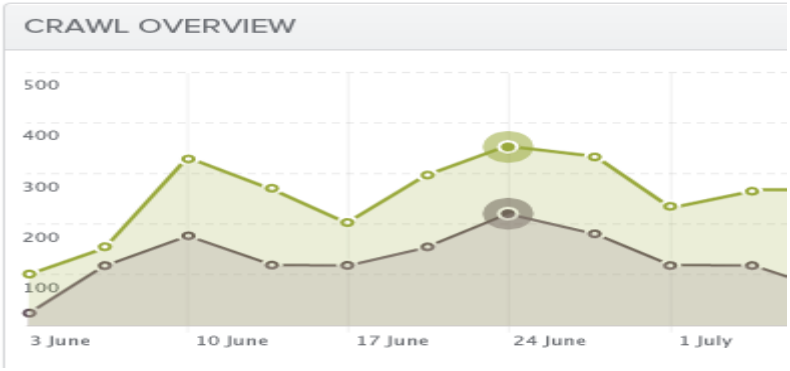

Figura 5 Indicador de Puntos de Tarea en Desarrollo
Fuente. PMOInformatica.com. (29 de Agosto de 2012). PMO Informática Oficina de proyectos. Obtenido de http://www.pmoinformatica.com/2012/08/5-metricas-para-proyectos-de-desarrollo.html

- **Gráfico Burdown**: Esta métrica de seguimiento relaciona el trabajo que queda por realizar en comparación con el tiempo que queda para realizarlo. Esta gráfica se da en cada iteración o sprint en función de las historias y tareas que se han incluido en la misma. Esta métrica es fundamental al momento de determinar si el sprint transcurre según lo planeado y si

logrará el equipo ágil cumplir con lo comprometido en la reunión de planificación.

Cálculo = Para cada punto de la gráfica se determina lo siguiente:

1. Puntos de historias que quedan por desarrollar y probar

2. Puntos que deberían quedar de acuerdo a la planificación del sprint. Para esto se debe calcular los Puntos totales planificados – Velocidad planificada en el sprint por los días transcurridos. La velocidad planificada se calcula mediante la fórmula: Puntos total planificados / Días de duración de Sprint.

3. Cada punto corresponde a las unidades previamente establecidas para las historias.

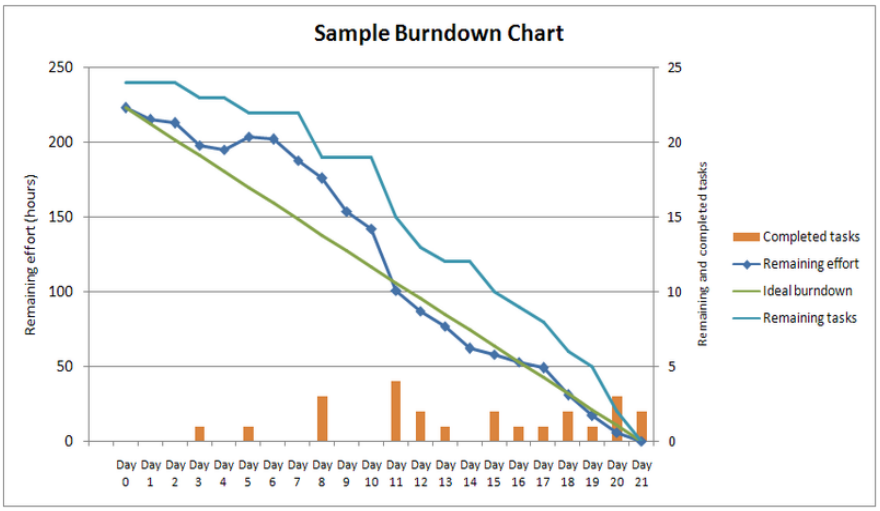

Figura 6 Gráfica Burdown
Fuente. PMOInformatica.com. (29 de Agosto de 2012). PMO Informática Oficina de proyectos. Obtenido de http://www.pmoinformatica.com/2012/08/5-metricas-para-proyectos-de-desarrollo.html

- **Velocidad**: Este indicador nos muestra el número de puntos de historia que están siendo completadas en una iteración por el equipo de desarrollo ágil. Generalmente a este indicador se lo suele medir al final de la iteración, sin embargo, puede ser útil para un día o momento específico. Normalmente a este indicador se lo considera dentro de la productividad del equipo, puesto que a un mayor valor en el cálculo, significa que existen tiempos inferiores para resolver las historias de usuario. Este indicador es el más común de la metodología SCRUM, y es bastante útil para tomar decisiones estratégicas cuando un proyecto tiene tiempos bastante ajustados. Al indicador se lo puede graficar en distintos momentos o comparar entre iteraciones para ver una velocidad promedio del equipo.

Velocidad = Suma de puntos de historia completados en la iteración

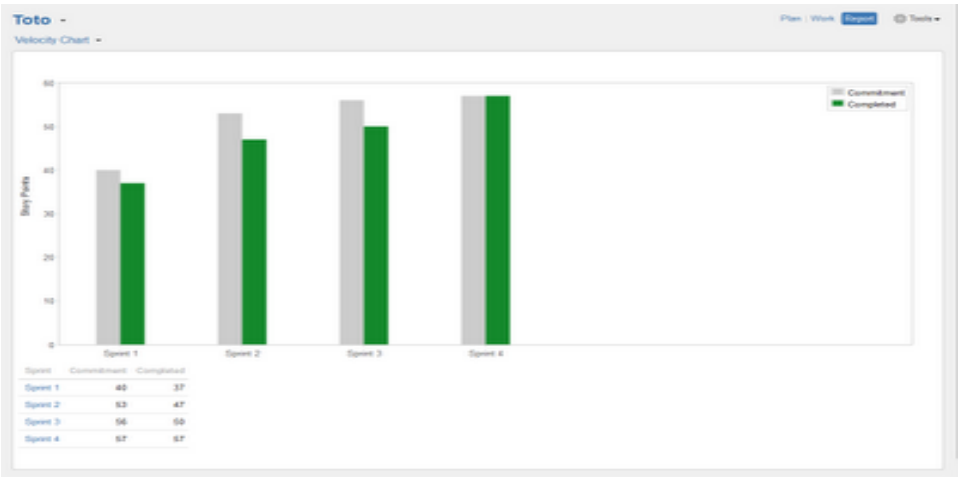

Figura 7 Velocidad de Equipo entre Sprints
Fuente. PMOInformatica.com. (29 de Agosto de 2012). PMO Informática Oficina de proyectos. Obtenido de http://www.pmoinformatica.com/2012/08/5-metricas-para-proyectos-de-desarrollo.html

- **Deuda Técnica**: Este indicador nos muestra el trabajo que debe hacerse, antes que una tarea sea definida como completa de acuerdo a la metodología. Muchas veces cuando los tiempos son bastantes cortos, los equipos de desarrollo optan por no optimizar el código, ocasionando que más tarde el equipo deba refactorizar varios puntos del mismo. Otro caso se da cuando no se cumplen a cabalidad los estándares de la empresa en el código fuente, y nos vemos obligados más tarde a refactorizarlos, a pesar de que se encuentre funcional. Un valor alto en este indicador nos muestra que la calidad del producto entregado no es la mejor. Este indicador puede ser comparado en las diferentes iteraciones para observar como varía la calidad del producto.

  Deuda Técnica = Valor total en puntos o porcentaje de la deuda técnica.

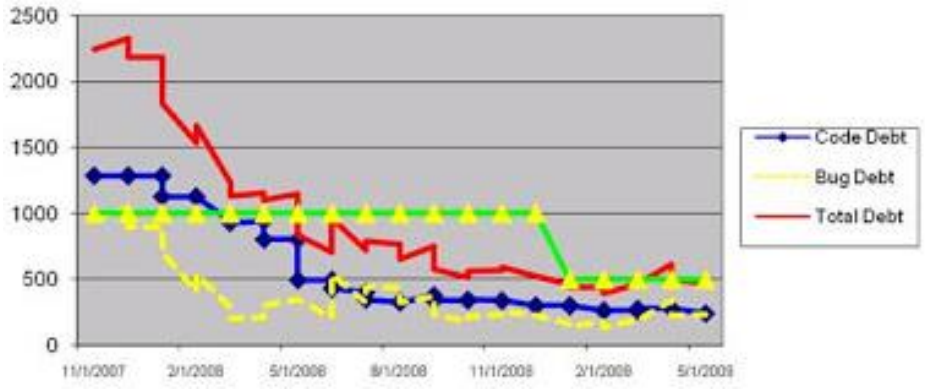

Figura 8 Deuda Técnica por Fechas
Fuente. PMOInformatica.com. (29 de Agosto de 2012). PMO Informática Oficina de proyectos. Obtenido de http://www.pmoinformatica.com/2012/08/5-metricas-para-proyectos-de-desarrollo.html

Siguiendo con el tema de la metodología, las principales ventajas de SCRUM son:

- Provee la facultad de adaptarse a cualquier contexto o área. Esta metodología puede usarse no exclusivamente para realizar software.

- Provee resultados de manera más rápida. Cada sprint o iteración sirve para entregar al cliente un desarrollo de inicio a fin. Es decir, cada entrega parcial genera rápidamente valor para el cliente.

- Administra las expectativas del Cliente. En cada sprint o iteración el cliente podrá dar una retroalimentación rápida sobre lo que espera del producto a recibir.

- Administración de Riesgos y Errores: Mediante la reunión de pie diaria o con la reunión de retrospectiva se descubren rápidamente los errores o riesgos que pueden afectar a un proyecto, y así poder tomar las acciones correctivas necesarias.

- Proporciona una mejor predicción de tiempos. Esta metodología permite conocer la velocidad media que tiene el equipo de trabajo por sprint, esto permite que sea más fácil estimar los tiempos de las tareas o funcionalidades pendientes en el backlog.

Las principales desventajas de la metodología SCRUM son:

- Se necesita un gran esfuerzo del equipo ágil para definir las tareas de la historia y sus tiempos. En caso de no realizar estas tareas correctamente la metodología pierde su esencia.

- Tiene mejores resultados en equipos de trabajo reducidos. Es mucho más manejable la metodología sobre equipo que no excedan 5 personas, debido a que las reuniones diarias deben ser rápidas. Por otro lado, los equipos deben ser auto-organizados y cuando se tiene grupos grandes se dificulta el conseguir esto.

- Se requiere que los miembros de los equipos ágiles tengan una buena experiencia sobre lo que están realizando. El éxito de esta metodología se apalanca en la experiencia. Scrum no se encuentra enfocado para equipos junior, que no conozcan lo que van a realizar.

Cabe tomar en cuenta que cuando se adopta SCRUM como metodología de desarrollo de software, suelen ocurrir errores comunes en la mayoría de la empresas.

2.2.4 Marco de Referencia Scaled Agile Framework SAFe

Dentro de la empresa donde se realizará el estudio se utiliza el Marco de Referencia denominado Scaled Agile Framework o por sus siglas SAFe. Este marco de referencia está basado en SCRUM y principios ágiles, para ayudar a ganar agilidad a la empresa a nivel de organización y ya no solo de equipo. Este marco de referencia fue creado por Dean Leffingwell y su principal

objetivo es incorporar un sistema con prácticas probadas e integradas con el fin de llevar mejoras sustanciales en el compromiso de los empleados, calidad de las soluciones, tiempo de comercialización y productividad dentro de los equipos de trabajo.

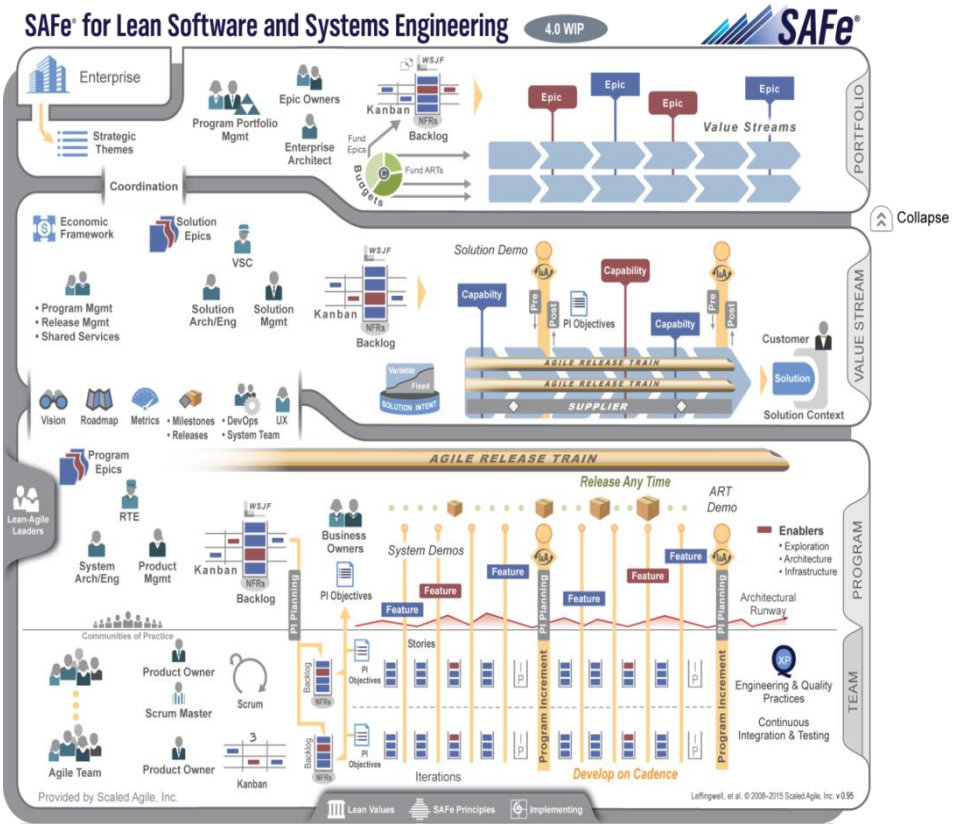

Figura 9 Scaled Agile Framework 4.0
Fuente. Leffingwell's, D. (13 de Junio de 2016). SAFe Scaled Agile Framework. Obtenido de http://www.scaledagileframework.com/

Según (Garzás, 2013), para la implantación de Scaled Agile Framework en una empresa se manejan 3 niveles de abstracción: nivel de portafolio, nivel de programa y por último nivel de equipo.

- **Nivel de Equipo:** En este nivel se organizan como trabajan los equipos ágiles que intervienen en el desarrollo de software. Scaled Agile

Framework propone que se maneje una combinación de técnicas de SCRUM con Extreme Programming XP. Como habíamos hecho mención antes sobre SCRUM, esta metodología se lleva a cabo con los roles de Product Owner, Scrum Master, y equipo ágil, para más tarde planificar el trabajo en iteraciones denominadas sprints. Esta planificación se la realiza con los insumos de historias de usuario dada por el Product Owner. Por otro lado, el Extreme Programing se da con historias especiales denominadas spikes. Estas historias tienen por característica fundamental la complejidad y el equipo ágil requiere investigación o familiarizarse con nuevas técnicas para su solución. Normalmente este tipo de historias no son atacadas por los módulos de la empresa en estudio, sino más bien por un equipo especializado en la búsqueda de nuevas tendencias y oportunidades de mercado.

Se sabe que lo único que varía en este nivel es el rol de Scrum Master, debido a que aparte de las funciones definidas por SCRUM, también deberá comunicar y coordinar el trabajo con otros Scrum Masters de otros equipos ágiles. Además, el Scrum Master tendrá que participar de las reuniones de ART Release Plannings, que se describirán en el siguiente nivel.

- **Nivel de Programa:** En este nivel se busca definir los Agile Release Train o ART, estos ART se los compara con las iteraciones a nivel de equipo pero a nivel de programa. Es decir, en el ART trabajan coordinadamente 5 o 10 equipos, sincronizando sus iteraciones y lanzamientos.

Además, en este nivel aparece el program backlog, y se refiere a una lista de features o características priorizadas por valor para la empresa. Normalmente esta lista es priorizada usando el Framework Weighted Shortest Job First o por sus siglas en inglés WSFJ. Este es un algoritmo de programación y priorización de trabajo para un ambiente donde la duración de los componentes de trabajo varía, así como también el coste de la demora. La fórmula de cálculo es la siguiente:

WSJF = (Valor de Negocio/Usuario + Valor del Tiempo + Reducción de Riesgo & Valor de Oportunidad) / Tamaño del trabajo

El resultado de dicho cálculo nos ayuda a priorizar las características, dando prioridad a las que poseen un valor más alto.

Siguiendo con el nivel de programa, los features o características del producto o servicio pueden generarse a nivel de programa o derivarse de épicas escogidas en el portafolio. Estas características a su vez pueden ser descompuestas en varias historias de usuario, que serán trabajadas en los sprints por el equipo ágil.

Es importante resaltar que en un lapso de 10 semanas o 5 iteraciones por lo general, se produce lo que llama el marco de referencia como un release o lanzamiento. Este lanzamiento busca un incremento potencialmente entregable o lo que es conocido en sus siglas en inglés como PSI.

Siguiendo con el marco de referencia SAFe, decimos que en este nivel hay un equipo de gestión de release denominado Release Management Team, donde se da la apertura a representantes de marketing, calidad y desarrolladores con el objetivo de aprobar lo que se va a entregar a los clientes en cada uno de los releases o lanzamientos.

Lo que se busca en este nivel de programa, es tener una visión más global sobre los objetivos a cumplirse en los ART, para de esta forma lograr coordinar a todos los equipos que trabajan en la empresa y tener los mismos objetivos.

Dentro de este nivel hacen su aparición nuevos roles del marco de referencia y son: System Architect, y Product Manager. El Product Manager es quién sabe qué cosas van en los ART, mientras que el System Architect es el encargado de dar lineamientos hacia la visión técnica, tecnológica y arquitectónica de la solución del producto o servicio, guiando a los equipos ágiles tanto en el diseño como la implementación del producto. También ayuda a construir lo que se conoce como la pista arquitectónica por donde circula el ART.

- **Nivel de Portafolio:** En este nivel se administran las épicas a alto nivel, en busca de ir en la dirección de los objetivos de la empresa y la arquitectura del sistema. Una épica es una historia más grande que una historia de usuario y sigue su mismo formato pero su alcance es mayor. Regularmente

las épicas se dividen en historias de usuario más pequeñas, debido a que la solución de la misma podría tomar más de una iteración. Lo que se realiza en este nivel es definir sin tanto detalle lo que más valor le da a la organización basados en los principios de Lean y la utilización de Tableros Kanban.

Un aspecto fundamental a analizar sobre Scaled Agile Framework son sus ventajas. Según (Barret, 2014) las principales ventajas son:

- **Promueve un pensamiento ágil en un mundo empresarial tradicionalmente alineado:** Scaled Agile Framework promueve un cambio organizacional basado en el liderazgo. Este liderazgo deberá desarrollar a las personas que participan en la empresa, descentralizará el control y se centrará en los principios del manifiesto ágil. Estos principios del manifiesto ágil son:

    1. Satisfacer al cliente mediante la entrega continua y rápida de partes de software útiles y de valor.

    2. Nuevos requisitos no son despreciados, incluso al crearlos en la fase final del desarrollo.

    3. Entrega de funcionalidad de software que funcionen correctamente, en lapsos de tiempo cortos. Se hablan de semanas y no meses.

4. La única prueba fehaciente para medir el avance es el software funcionando.

5. Logra un desarrollo a ritmo constante, es decir sostenible en el tiempo de desarrollo.

6. Se trabaja a diario muy estrechamente entre los desarrolladores y personas de negocio.

7. La conversación de persona a persona es la mejor forma de comunicación.

8. Los proyectos se construyen alrededor de personas motivadas, para lo cual se les entrega la confianza necesaria al realizar una tarea.

9. Simplicidad

10. Equipos auto organizados.

11. Adaptación a los cambios.

12. Se da especial atención al buen diseño y a la excelencia técnica dentro de los proyectos.

- **Utiliza combinación de técnicas:** Scaled Agile Framework usa una combinación de técnicas ágiles como desarrollo guiado por pruebas, procesos formales de revisión ente pares, integración continua, y refactorización.

- **Disminuye el tiempo de Time to Market:** De acuerdo a la utilización de Scaled Agile Framework en empresas, se ha visto que disminuye entre un 30% a un 75% el tiempo de Time to Market.

- **Incrementa la productividad:** Scaled Agile Framework asegura que la productividad en las empresas que lo usen, aumentará entre un 20% a un 50%.

- **Disminuyen los defectos de Producción:** La experiencia de implantación de Scaled Agile Framework en empresas ha demostrado que reducen en un 50% los errores que se dan en producción.

2.3 FUNDAMENTOS TEÓRICOS

2.3.1 Enfoque basado en Procesos

Tomando en cuenta que la empresa donde se quiere mejorar el proceso de desarrollo de software en banca en línea tiene certificación ISO 9001 nos basaremos en el enfoque que le da la misma.

Para que una organización funcione de manera eficaz, tiene que determinar y gestionar numerosas actividades relacionadas entre sí. Una actividad o un conjunto de actividades que utiliza recursos, y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados, se puede considerar como un proceso. Frecuentemente el resultado de un proceso

constituye directamente el elemento de entrada del siguiente proceso. La aplicación de un sistema de procesos dentro de la organización, junto con la identificación e interacciones de estos procesos, así como su gestión para producir el resultado deseado, puede denominarse como "enfoque basado en procesos". Una ventaja del enfoque basado en procesos es el control continuo que proporciona sobre los vínculos entre los procesos individuales dentro del sistema de procesos, así como sobre su combinación e interacción (ISO 9001, 2008).

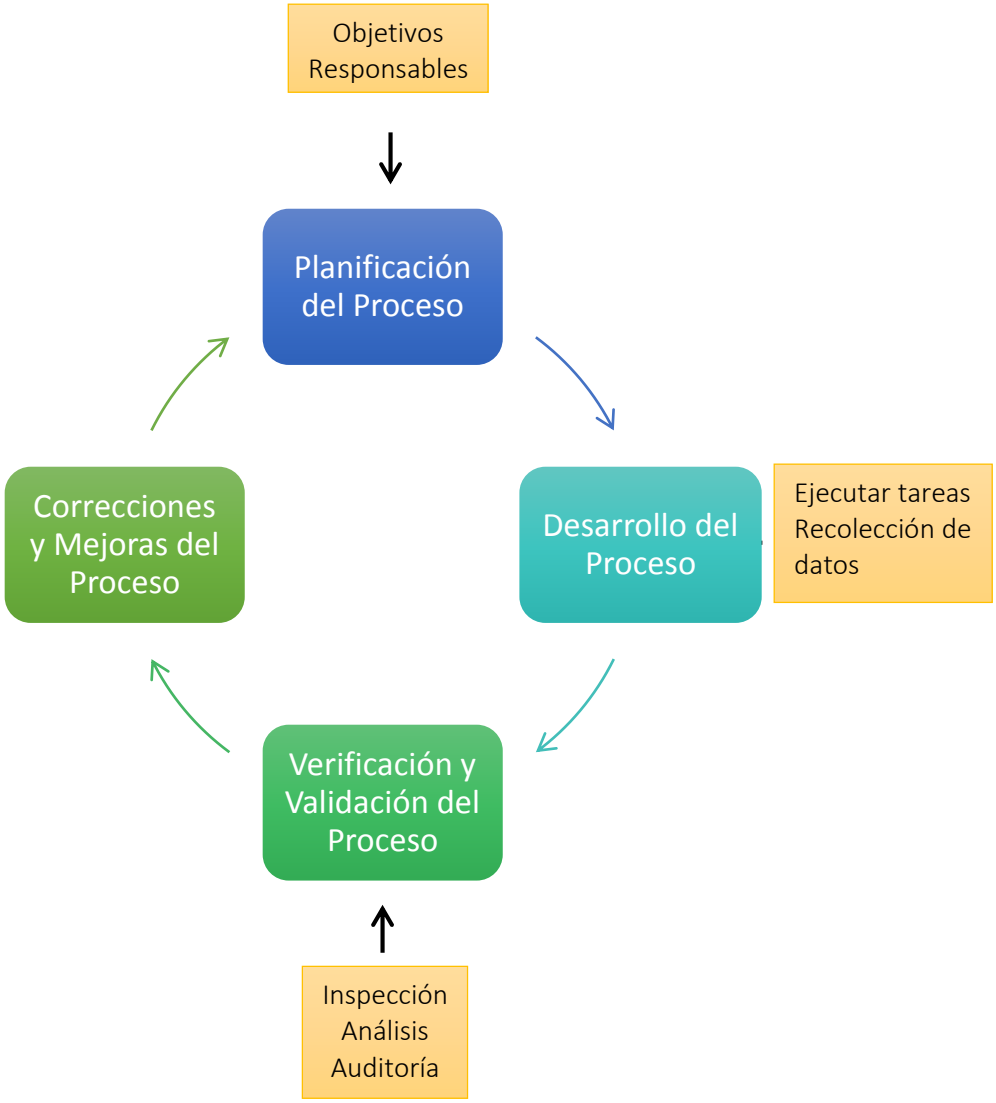

Figura 10 Ciclo de Mejora Continua del Proceso
Fuente. Gómez, N. (27 de Junio de 2009). Calidad y gestión empresarial. ISO 9001 e ISO 14001. Adaptado de http://hederaconsultores.blogspot.com/2009/06/enfoque-procesos-principios-iso-9001.html

2.3.2 Fases de mejora de un proceso

De acuerdo con (Maldonado, 2013) la metodología para una correcta mejora de procesos basados en ISO es la siguiente:

1. **Definir los límites del proceso:** Básicamente se definirá el inicio y el final del proceso, así como también se identificará los insumos y rendimientos del proceso.

2. **Observar los pasos del proceso:** Cuando las personas describen un proceso sin observarlo en realidad, casi siempre dejan algunas cosas fuera. Dentro de este paso se deberían haber observado todos los pasos del proceso, registrando todos los pasos del mismo, identificado el flujo y secuencia del proceso y por último clasificando todos los tipos de pasos del proceso.

3. **Recabar los datos relativos al proceso:** Aquí se tendrán datos cuantitativos como tiempo, número de personas, distancia y cantidad de defectos relacionados con el proceso.

4. **Analizar los datos recabados:** Generalmente los problemas evidentes surgen sin tener que realizar muchos cálculos o análisis. Se obtiene poco al refinar éstos en forma continua.

5. **Identificar las áreas de mejora:** El objetivo primordial de este paso es eliminar o reducir al mínimo el desperdicio, enfocándonos generalmente en transporte, demoras, inspección, retrabajo y almacenaje. Cuando se eliminan o reducen al mínimo estos pasos, es posible comenzar a mejorar los pasos de operación. Es decir enfocaremos nuestros esfuerzos en los siguientes puntos: pasos de transporte redundantes o innecesarios, pasos de transporte que consumen tiempo, pasos de demora redundantes o innecesarios, pasos de demora que consumen tiempo, pasos redundantes de inspección, todos los pasos de retrabajo, diagramas ineficientes de proceso, secuencias o flujos de proceso ineficientes.

6. **Desarrollo de mejoras:** Aquí se buscará desarrollar una mejora apropiada, basada en eliminar varios pasos del proceso, en especial los que no le agregan valor, reducir al mínimo el tiempo asociado con ciertos pasos, reducir la complejidad del proceso al simplificar éste, combinar varios pasos de proceso, elegir un método alterno de transporte, cambiar un proceso lineal a paralelo, usar rutas alternas de proceso que se basan en decisiones, cambiar la secuencia de pasos del proceso, usar la tecnología para elevar la eficacia o eficiencia del proceso, y dejar que los clientes hagan algo del trabajo del proceso.

7. **Implantar y vigilar las mejoras:** Este paso buscará implantar las mejoras de la siguiente forma: una corrida piloto, un cambio completo o un cambio gradual. La forma en la que se elija implantar la solución dependerá del costo, complejidad y riesgo al fracaso.

2.3.3 Mejora continua de procesos

Según la norma (ISO 9001, 2008) la mejora continua es mejorar la eficacia de su sistema aplicando la política de calidad, los objetivos de calidad, los resultados de las verificaciones de inspección, el análisis de los datos, las acciones correctivas y preventivas y la revisión de la Dirección. La organización debe identificar de qué manera los procesos citados contribuyen a la mejora constante del Sistema de Gestión de Calidad. Además el punto clave consiste en revisar la correlación entre estos procesos, asegurándose que contribuyan conjuntamente a la mejora constante. Los datos de un proceso deben analizarse y convertirse en datos preliminares para otro proceso que a su vez dará lugar a una acción para corregir o mejorar el Sistema de Gestión de Calidad

Actualmente la empresa maneja una certificación de norma ISO 9001-2008, misma que da la pauta a la empresa a dar un giro hacia procesos más controlados y que se mejoren día tras día.

2.3.4 Herramientas para la mejora continua de procesos.

Según (González, 2012), el mayor enemigo de los procesos se da por la variabilidad, misma que se puede ver en características cuantificables de productos y servicios. La variabilidad en los procesos es la preocupación de toda empresa debido a que existe en todas las etapas del ciclo de vida de los productos.

Cuando se tienen tareas estandarizables en el proceso, es decir tareas repetitivas y con tiempos definidos, las técnicas estadísticas como el análisis de correlación y el histograma, nos pueden proveer las causas, extensión y naturaleza de dicha variabilidad. Sin embargo, en procesos de desarrollo de software donde no se tienen tareas estandarizables podemos usar herramientas como:

- **Diagrama de Análisis Causa Efecto:** De acuerdo con (González, 2012) este diagrama sirve para identificar los diferentes tipos de causas que actúan sobre un problema. Comúnmente se los conoce como diagrama de Ishikawa o diagrama de espina de pescado. Normalmente este diagrama categoriza a las causas en las categorías de métodos, mano de obra, maquinaria, materiales y medio ambiente, sin embargo para poder contextualizar mejor los problemas en el desarrollo de software, se ha decidido categorizar en personas, tecnología, metodología, y medición. Los pasos a seguir para realizar un diagrama de este tipo son:

  1. Se identifica el problema que representa la cabeza de pescado.

  2. Se determina las causas tomando en cuenta las categorías dadas.

  3. Mediante el equipo de trabajo se realiza una lluvia de ideas sobre las causas del problema.

4. Se identifica en el diagrama las causas que más se repiten y se las prioriza por su recurrencia.

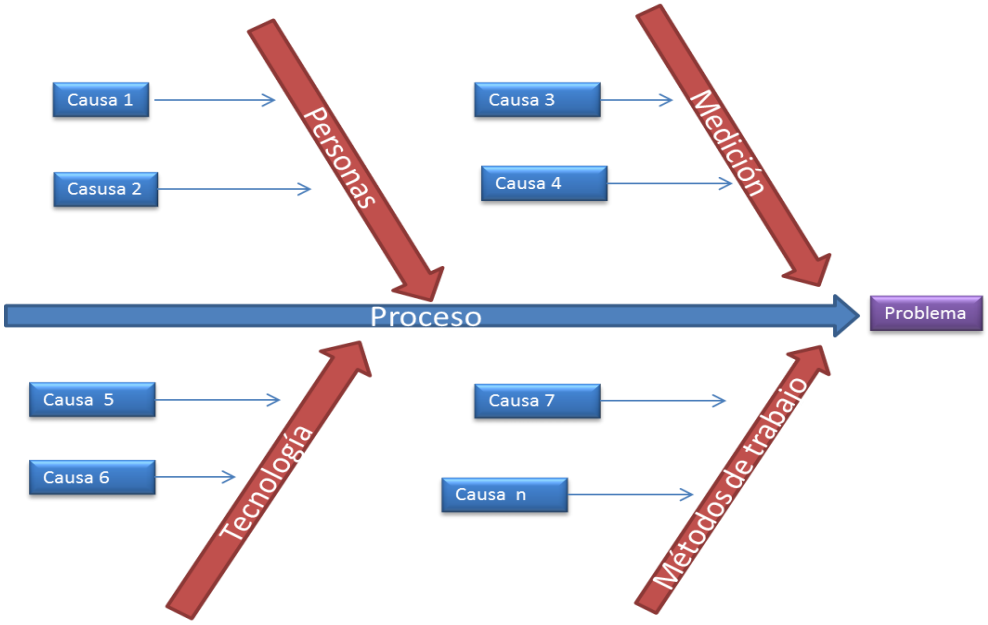

Figura 11 Ejemplo Diagrama de Análisis de Causa y Efecto
Fuente. Empresa Desarrolladora de Software analizada, 2016

- **Lluvia de Ideas:** Esta es una herramienta útil para identificar objetivos, problemas actuales, y problemas potenciales al realizar algún trabajo. Normalmente se la usa para obtener información de un proceso y al mismo tiempo crear entusiasmo y un ambiente de participación. Esta herramienta forma parte de una de las fases del diagrama de causa-efecto y para su correcta realización se requiere respetar las siguientes reglas:

1. La participación debe ser por parte de todo el grupo.

2. Se deben escribir todas las ideas en un pizarrón o en algún lugar donde todo el grupo la pueda ver.

3. No se deben identificar culpables, ni existen ideas incorrectas.

3. ESTADO ACTUAL DEL PROCESO DE DESARROLLO DE SOFTWARE EN EL MÓDULO DE BANCA EN LÍNEA

3.1 METODOLOGÍA Y HERRAMIENTAS PARA EVALUAR EL PROCESO

La metodología que se decidió usar para evaluar al proceso de desarrollo de software y poder realizar una propuesta es la metodología de los 7 pasos o más conocido como método MP. Esta metodología nos provee una forma sistémica para analizar los procesos y busca obtener resultados cuantificables en el tiempo. Además nos ayuda a identificar los problemas y puntos de mejora del mismo. Esta metodología se basa en el enfoque de procesos de ISO y sus 7 pasos son:

1. Definir los límites del proceso.

2. Observar los pasos del proceso.

3. Recolectar los datos relativos al proceso.

4. Analizar los datos recolectados.

5. Identificar las áreas de mejora.

6. Desarrollar mejoras.

7. Implantar y vigilar las mejoras.

Cada paso de la metodología fue detallado en el marco teórico y se aclara que no se aplicó el séptimo paso de la metodología porque este trabajo de titulación no abarca la implantación de las mejoras.

Para poder realizar tanto el primero como el segundo paso de la metodología, es decir, definir los límites del proceso y poder observar los pasos del mismo, se ha usado la herramienta de diagrama de flujo en formato vertical. Esta herramienta permite una representación gráfica y estándar de cada paso del proceso conjuntamente con los correspondientes actores de la misma. La gran ventaja que alcanzamos con esta herramienta es la comprensión rápida y ágil del proceso.

Siguiendo la metodología, para recolectar y analizar los datos del proceso, se usó la herramienta denominada hojas de verificación. Este tipo de hojas, proporcionan un medio para registrar efectiva y eficientemente datos que nos servirán para un análisis subsecuente. Se realizó una lista de verificación donde se tomaban en cuenta los tipos de errores que se dan en el equipo de trabajo por sprint y se los totalizaba por proyecto. A cada uno de estos errores, se les registraba el tiempo que tomaban en resolverse. Esta lista de verificación no se adjunta al trabajo de titulación por motivos de política de seguridad de la empresa, sin embargo se los expresa de manera clara en el punto “Recolección de información e identificación de puntos de mejora del proceso” desarrollado más tarde. Además, se recopilan

datos en base a indicadores propios del proceso. Los tres indicadores que se toman en cuenta son: el porcentaje de cumplimiento por cada sprint del equipo ágil de desarrollo, la deuda técnica del equipo por sprint y el número de errores cometidos por sprint. Es importante resaltar, que por motivos de políticas de seguridad de la empresa, no se han tomado en cuenta otros indicadores en valor como costo unitario de producción, costo por horas hombre en sprint, productividad del equipo por sprint, eficiencia del equipo, valor del negocio entregado al cliente. Sin embargo, mi propuesta se ha enfocado en disminuir el tiempo en los pasos del proceso de desarrollo de software que afectan directamente a la productividad y eficiencia del equipo ágil como vemos en sus indicadores.

Por otro lado, en la recolección de datos existió el problema que hasta ese momento solo se había realizado un proyecto que nos provean los indicadores de este proceso de desarrollo de software ágil en el módulo de banca en línea, debido a que los proyectos de software normalmente toman entre semestres y años su desarrollo, y este proceso es relativamente nuevo en la empresa de desarrollo de software. Otro inconveniente es que no se puede tomar los datos de otros módulos o áreas, debido a que cada una de ellas posee su propia tecnología de desarrollo de software, se enfrentan a otras realidades y problemas, a pesar de compartir el mismo proceso de desarrollo de software ágil. Por estas razones, la propuesta de mejora del proceso de desarrollo de software del módulo de banca en línea basó sus datos en un estudio de Caso Único. La metodología de caso único tiene las ventajas de poder generar hipótesis, nos provee la capacidad de establecer las condiciones y naturaleza de una relación entre causa-efecto, la capacidad de trabajar en situaciones atípicas por motivos como los descritos antes de falta de datos de más

proyectos, y por último nos provee un valor persuasivo, pudiendo convertir en ideas concretas, conceptos considerados como principios abstractos. En cambio, las desventajas de este método de caso único son: la no posibilidad de generalizar sus propuestas o conclusiones, porque estadísticamente se necesita un muestreo mayor que provea más confiabilidad en las mismas, y la variabilidad por diversos factores de los proyectos dentro de la empresa de software pueden ocasionar cierto error en las inferencias del estudio.

Para el quinto paso de la metodología, que se refiere a identificar las áreas de mejora, se usó el diagrama de causa y efecto o diagrama de Ishikawa. Se decidió el uso de esta herramienta debido a que las tareas de desarrollo de software no tienen tiempos estandarizables, definidos y comparables fácilmente entre proyectos. Por ejemplo al desarrollar una página web de inicio de sesión, los desarrolladores deben evaluar si tiene servicios web, si existe migración de información, la tecnología sobre la cual se realizará o reutilizará, los campos a validarse, etc. Es decir, los requerimientos normalmente los pone el cliente conjuntamente con las leyes del país y son bastante variables, con lo cual se da como resultado que no se tenga tareas comparables con facilidad. Normalmente, otras herramientas más estadísticas son usadas para el análisis de los procesos, como por ejemplo el histograma y gráficas de control, sin embargo, el desarrollo de software al no tener tareas estandarizables o con tiempos definidos comparables, no nos permite realizar un análisis comparativo estadístico de manera natural. Es importante resaltar que el diagrama de causa y efecto nos provee la facilidad de participación de todo el equipo ágil para identificar los principales problemas conjuntamente con sus causas mediante el uso de lluvia de ideas, y al ser bastante gráfica facilita el entendimiento

de sus causas categorizadas en personas, tecnología, metodología, y medición. Una vez se ha identificado los problemas con sus respectivas causas en el proceso, se evaluará la madurez del proceso usando el modelo de madurez CMM.

Para el sexto paso de la metodología, que se refiere al desarrollo de la mejora, se usarán los problemas encontrados con el diagrama de causa y efecto y se definirán las soluciones a las mismas mediante el análisis de las causas identificadas. Además, se realiza un análisis de priorización de implementación de las soluciones propuestas mediante el uso de una Matriz de impacto y facilidad de implementación. Esta matriz permite priorizar el orden de las soluciones dadas en esta propuesta para que tengan un mayor impacto en el menor tiempo.

3.2 DEFINICIÓN DE LOS LÍMITES DEL PROCESO

Cuando se requiere una nueva funcionalidad para un cliente, se inicia con el levantamiento de requerimientos por parte de un especialista funcional del módulo, más tarde el Product Owner conjuntamente con el apoyo técnico del System Architect elaboran una propuesta, dicha propuesta es cuantificada en horas hombre de trabajo y en un valor monetario. Una vez el cliente acepta la propuesta entregada y firma un contrato de trabajo, comienza el proceso a mejorar en esta propuesta, conocido como proceso de desarrollo de software. Este proceso de desarrollo de software es compartido para todas las áreas y módulos de la empresa. Cabe indicar que la elaboración de la propuesta no conforma parte del proceso de desarrollo de software, pero sirve para ponernos en contexto.

El proceso de desarrollo de software inicia con la etapa de catálogo o portafolio de productos y mejoras, donde básicamente se busca y define una versión del producto basados en la visión estratégica de la empresa que servirá como base del desarrollo de software. Normalmente la visión estratégica busca tener un producto reutilizable en su mayoría y con la última tecnología disponible que permita tener una excelente experiencia de usuario. Más tarde, a la versión del producto se le realizarán los cambios personalizados de acuerdo al cliente. El product Manager, Product Owner y System Architect definen las características del software a desarrollar, las estiman en cuanto a esfuerzo en horas y las priorizan. Siguiendo el flujo se definirá un equipo de programadores denominado equipo ágil, y se descompondrán las características definidas por el Product Owner, Product Manager y System Architect, en historias de usuario que tendrán una estimación de esfuerzo en horas y a las cuales el equipo deberá comprometerse de acuerdo a su capacidad a ir entregando en entregas parciales denominadas Sprints. Estas entregas parciales normalmente son cada dos semanas y sirven para que el cliente vea como avanza el producto. Cada cierre de Sprint el equipo deberá hacer una entrega de historias de usuario que puedan ser probadas por el Product Manager y el Cliente en un ambiente interno de pruebas. Por último, se consensuará con el cliente la entrega de funcionalidades completas que se certificarán mediante pruebas en ambientes del cliente. Es importante resaltar, que cuando se ha certificado el software por parte del cliente, existirá una etapa de puesta en producción tomando en cuenta las necesidades del cliente en cuanto a tiempo. El proceso de desarrollo de software concluye una vez se certifica el software por el cliente y es puesto en producción.

3.3 PASOS DEL PROCESO

El proceso de desarrollo de software del módulo de Banca en Línea se detalla así:

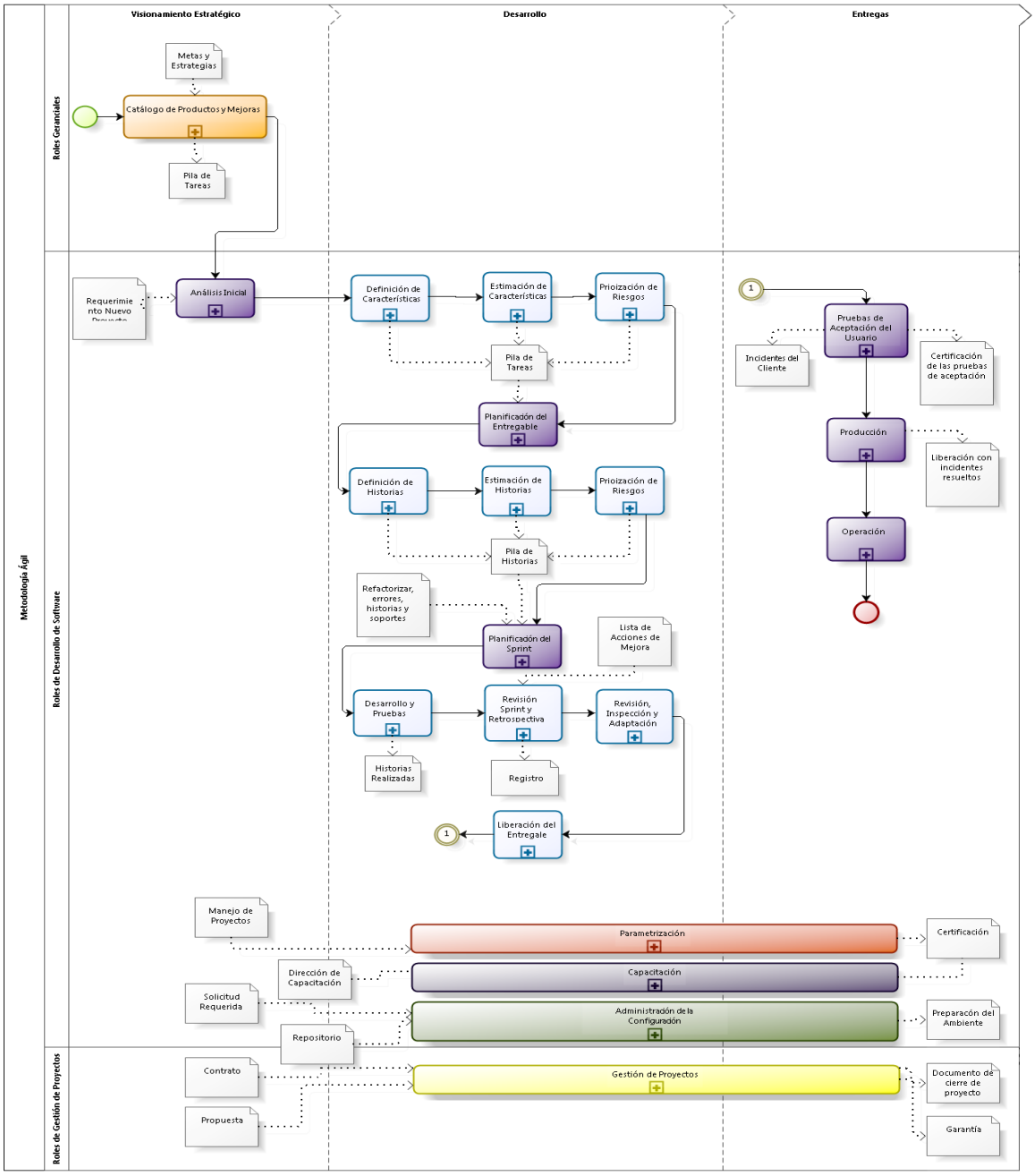

Figura 12 Proceso Ágil de Desarrollo de Software Módulo Banca en Línea
Fuente. Daniel Jarrín, 2016

Antes de comenzar el desarrollo de software como tal, la empresa con sus roles gerenciales definen un catálogo de productos que se usarán como base para el

desarrollo de las características que se definirán más tarde. Mediante un análisis del proyecto en cuanto a objetivos y la orden de trabajo, la primera parte del proceso comienza enfocándose en la definición de características del software conjuntamente con la planificación, priorización y estimación del tiempo que toma realizar dichas características. Estas características son definidas por el Product Owner, Product Manager, y con el apoyo técnico del System Architect. El resultado será una pila de características o tareas. Las priorización de la pila de características se realizarán usando el algoritmo Weighted Shortest Job First o WSFJ, donde participará el cliente constantemente. Este algoritmo se explicará a detalle en la descripción a del proceso que se encuentra en los siguientes numerales. Más tarde, cada una de las características se dividirá en historias de usuario con criterios de aceptación que permiten entregas parciales y de calidad. Estas historias son escritas a detalle por el Product Owner en base las características, para más tarde ser validadas por el Product Manager y el System Architect en cuanto a criterios de aceptación, funcionalidad, esfuerzo requerido y requerimientos técnicos. Cuando las historias han sido definidas y aceptadas, estas se priorizarán en cuanto a su entrega en el tiempo para estar acorde a la priorización de las características del software. Esta primera parte es desarrollada por lo que es conocido en el Scaled Agile Framework como nivel de programa.

La segunda parte se enfoca en el nivel de equipo, y es donde el marco de trabajo Scaled Agile Framework SAFe se relaciona con SCRUM. Aquí el equipo de trabajo toma la pila de historias priorizadas por el nivel de programa y se comienza a planificar en sprints el desarrollo de las mismas. Para planificar el Sprint se reúnen el Scrum Master, el Equipo Ágil de desarrollo, el System Architect, y descomponen las

historias de usuarios en tareas mediante el uso del Planning Poker que se explicará a detalle en la descripción del proceso. Básicamente, en la planificación el equipo se compromete a entregar n puntos de historia en el lapso de tiempo de un sprint. Siguiendo el flujo, el proceso de desarrollo de software y pruebas de las tareas que a su vez conforman las historias, se desarrollan a diario y el desarrollador de software del equipo ágil se compromete a informar el estado de cada tarea mediante el uso de un tablero Kanban con estados como: por hacer, en progreso, terminada, en pruebas y aprobada. Para que una historia pase a estado de pruebas y el tester cumpla su responsabilidad, el desarrollador debe cumplir todos los criterios de aceptación de la historia y a su vez resolver todas las tareas que componen la historia. Una vez, el tester haya aprobado las historias se debe enviar para la revisión e inspección de las mismas al Product Owner y Product Manager. Estos dos roles tendrán la potestad de devolver alguna historia que no cumpla lo que se pedía en las características o de lo contrario aprobarlas para la revisión del cliente en la liberación del entregable. Cabe anotar, que el equipo de desarrolladores conjuntamente con el Product Owner y System Architect, deberán realizar la reunión de entrega de la demo cada fin de sprint, antes de la liberación como tal y se deberá realizar las reuniones de retrospectiva que deben dejar por escrito un documento con lo que se hizo bien y mal en el sprint.

La tercera parte del proceso comprende a la entrega de la funcionalidad al cliente, y donde cada funcionalidad entregada por el equipo de desarrollo ágil deberá pasar por las pruebas de certificación del cliente y donde se genera un documento de la misma, así como también se registran las incidencias y la solución de ellas en caso de darse.

Además el proceso de desarrollo de software muestra las dependencias con otros procesos de apoyo de otras áreas como por ejemplo la administración de la configuración, parametrización, gestión de proyectos, etc.

El proceso de desarrollo de software se relaciona con el proceso de parametrización cuando para comenzar un desarrollo se requiere que el ambiente posea información del cliente en cuanto a cuentas, costos de transferencias, etc. Para esto inicialmente se debe elaborar un plan de parametrización y terminará con una certificación de la parametrización por parte del cliente y el equipo de desarrollo de software que usa esa información.

El proceso de desarrollo de software se relaciona con el proceso de administración de la catalogación con dos partes. La primera se da cuando el equipo de desarrollo de software ágil desea enviar a pasar sus desarrollos a un ambiente denominado de pruebas, que es donde tanto el tester, Producto Owner, Product Manager y cliente prueban la funcionalidad desarrollada. Aquí se crea un documento de catalogación y se da por terminada una vez se ha verificado la correcta funcionalidad en pruebas. La segunda se da cuando se requiere preparar un ambiente tanto de desarrollo como de pruebas. Se entiende como un ambiente al hardware y software que se necesitan para que un desarrollo funcione correctamente. El paso para la creación de un ambiente empieza con la solicitud de la misma, y termina con la certificación por parte del equipo de desarrollo en cuanto a su correcto funcionamiento.

El proceso de desarrollo de software se relaciona con el proceso de capacitación cuando el cliente se encuentra en sus pruebas de certificación y requiere el

conocimiento tanto técnico como funcional para armar su documento de pruebas. Normalmente comienza con un plan de capacitación y se termina con la certificación de dicha capacitación por parte del cliente.

Por último, el proceso de desarrollo de software se relaciona con el proceso de gestión de proyectos, debido a que este proceso permite gestionar los objetivos del proyecto y los riesgos del mismo cuidando las variables principales del proyecto: costos, tiempos, calidad y alcance. Se ejecuta cuando un proyecto ha sido aprobado por la empresa y por lo tanto se ha asignado un presupuesto, un Gerente de Proyecto y recursos necesarios. Normalmente, dan seguimiento a los indicadores por sprint del proceso de desarrollo de software, para evidenciar retrasos y tener posibles acciones a tomar.

Tabla 3:
Roles involucrados en el Proceso de Desarrollo de Software en el módulo de Banca en Línea

Roles Gerenciales	Roles de Desarrollo de Software	Roles De Gestión de Proyectos
• Equipo de Gestión de Portafolio (Portfolio Management Team)	• PM Project Manager • PO Product Owner • System Architect • Tester • Catalogador • Equipo Ágil • Administración de la Configuración • Administrador de ambientes • Administrador de Repositorio • Soporte Nivel 1/Soporte TI • Cliente • Stakeholders	• Sponsor del Proyecto • Cliente

Tabla 4:
Entradas y Salidas del Proceso de Desarrollo de Software en el módulo de Banca en Línea

Resumen entradas	Resumen Salidas
• Metas y Estrategias • Requerimiento Nuevo Proyecto • Pila de Historias • Refactorizar, errores, historias y soportes • Lista de Acciones de Mejora • Cronograma General • Manejo de Proyectos • Dirección de Capacitación • Solicitud Requerida • Repositorio • Contrato • Propuesta	• Pila de Tareas • Pila de Historias • Registro • Retrospectivas y Acciones de mejora • Historias Realizadas • Certificación de las pruebas de aceptación • Incidentes del Cliente • Liberación con incidentes resueltos • Certificación • Preparación del Ambiente • Documento de cierre de proyecto • Garantía

3.4 CARACTERIZACIÓN DEL PROCESO

Como se mostró en el diagrama de flujo del proceso de desarrollo de software ágil, los involucrados y el resumen de entradas y salidas del proceso, éste es general para toda la empresa y se lo usa alrededor de todas las áreas y módulos de la empresa.

PROCESO DE DESARROLLO DE SOFTWARE ÁGIL	
CARACTERIZACION DEL PROCESO DE DESARROLLO DE SOFTWARE ÁGIL	
TIPO DE PROCESO:	Desarrollo Core de Negocio
OBJETIVO:	Proveer el procedimiento y administración de recursos para un desarrollo ágil de proyectos de software
RESPONSABLE:	Gerente de Servicios de Ingeniería y Desarrollo

ALCANCE

El proceso de desarrollo de software inicia con la etapa de catálogo o portafolio de productos y mejoras, donde básicamente se busca y define una versión del producto basados en la visión estratégica de la empresa que servirá como base del desarrollo de software. Más tarde, a esta versión se le realizarán los cambios personalizados de acuerdo al cliente. El product Manager, Product Owner y System Architect definen las características del software a desarrollar, las estiman en cuanto a esfuerzo en horas y la priorizan. Siguiendo el flujo se definirá un equipo de programadores denominado equipo ágil, y se descompondrán las características definidas por el Product Owner, Product Manager y System Architect, en historias de usuario
que tendrán una estimación de esfuerzo en horas y a las cuales el equipo deberá comprometerse de acuerdo a su capacidad a ir entregando en entregas parciales denominadas Sprints. Estas entregas parciales normalmente son cada dos semanas y sirven para que el cliente vea como avanza el producto. Cada cierre de Sprint el equipo deberá hacer una entrega de historias de usuario que puedan ser probadas por el Product Manager y el Cliente en un ambiente interno de pruebas. Por último, se consensuará con el cliente la entrega de funcionalidades completas que se certificarán mediante pruebas en ambientes del cliente. Es importante resaltar, que cuando se ha certificado el software por parte del cliente, existirá una etapa de puesta en producción tomando en cuenta las necesidades del cliente en cuanto a tiempo. Este proceso de Desarrollo de Sofware es compartido por todas las áreas y módulos de la empresa cuando se tiene el requerimiento de un desarrollo.

PROVEEDORES	ENTRADAS	Subprocesos	Procesos Apoyo Areas relacionadas	SALIDAS	USUARIOS
Equipo de Gestión de Catálogo de Productos	Metas y Estrategias del área	Catálogo de Productos y Mejoras	Parametrización (área en cuestión)	Pila de Tareas o Carcterísticas priorizadas	Cliente
	Requerimiento Nuevo Proyecto	Análisis Inicial	Capacitación (área en cuestión)	Pila de Historias de Usuario priorizadas	Stakeholders
	Producto base para arrancar Desarrollo	Definición, priorización características	Administración de configuración. Catalogación y Ambientes (área de Administración y Configuración)	Historias Realizadas	
		Planificación entregable		Registro de tareas	
		Definición, Estimación Historias		Retrospectivas y acciones de Mejora	
		Planificación Sprint		Incidentes del cliente	
		Desarrollo, Pruebas	Gestión de Proyectos (área de Administración de Proyectos)	Documento de Certificación cliente	
		Retrospectiva		Documento de cierre y garantía proyecto	
		Liberación Entregable			

INDICADORES

TIPOS	INDICADORES
Metodología SCRUM	Velocidad de Equipo en puntos, % Cumplimiento Sprint, Valor de Negocio entregado al cliente en puntos y % de Deuda Técnica
Productividad	Costo unitario de Producción, Productividad por Sprint
Efectividad	% de cumplmiento de Trabajo Objetivo o Meta
Eficiencia	Eficiencia por sprint, Costo por Horas Hombre Sprint
Errores	Número de errores cometidos por sprint

RECURSOS

HUMANOS	TECNOLÓGICOS E INFRAESTRUCTURA
Product Manager PM, Product Owner, Equipo Ágil, Tester, Catalogador, Administrador de ambientes, Espacialistas de centro de soporte, Administrador de repositorios	Plataforma donde se registran características, historias y tareas del software con estimación planificada y real.

Figura 13 Caracterización Proceso Ágil de Desarrollo de Software Módulo Banca en Línea
Fuente. Daniel Jarrín, 2016

3.5 DIAGRAMA DE RELACIÓN DEL PROCESO

Como se mostró en el diagrama de flujo, el proceso de desarrollo de software se relaciona con procesos de apoyo y áreas que permiten o ayudan a completar las distintas actividades del mismo. Aquí tenemos el detalle de cómo se relacionan y en qué momento.

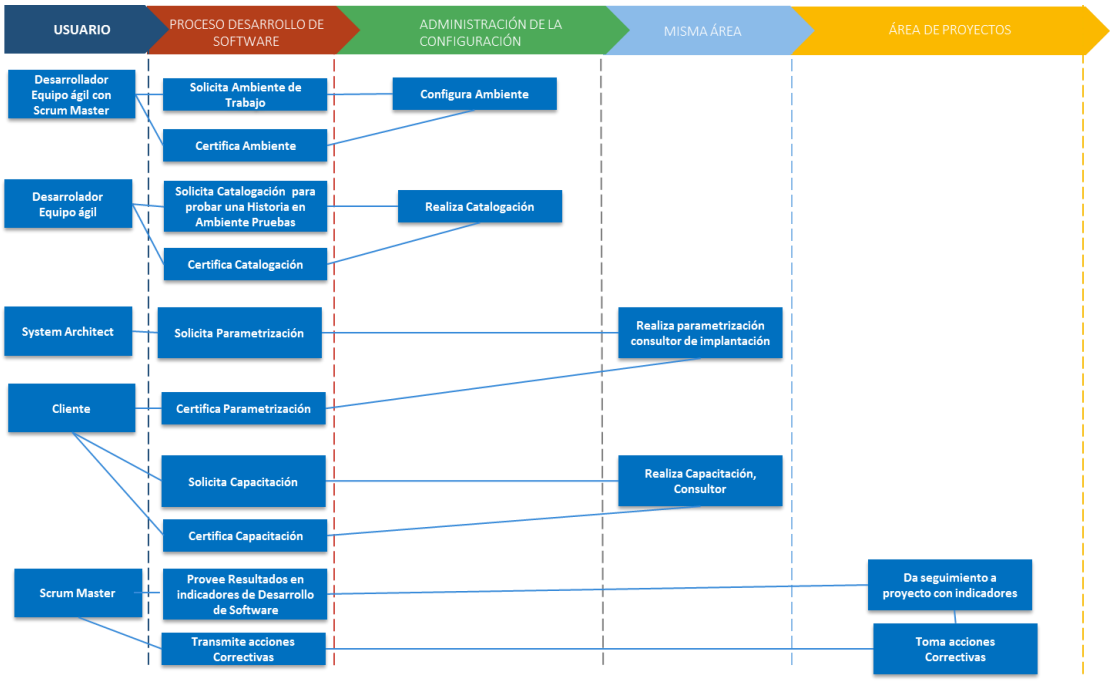

Figura 14 Diagrama de relación del proceso de desarrollo de software
Fuente. Daniel Jarrín, 2016

3.6 DESCRIPCIÓN DEL PROCESO

A continuación se describe cada uno de las subprocesos del proceso de desarrollo de software del módulo de Banca en Línea:

1. **Catálogo de Productos y Mejoras:** Esta subproceso se compone del catálogo Estratégico y Catálogo de aprobación de Épicas. El objetivo del

Catálogo Estratégico es evaluar las estrategias organizacionales, en función del catálogo de productos. Dentro del catálogo estratégico se definen los objetivos que se tienen con el producto por parte de la gerencia y el producto Manager, en busca de poder usar el producto para la mayor cantidad de ventas y sea lo suficientemente genérico, reutilizable. mantenible y tecnológicamente adaptable. Por otro lado, el objetivo del Catálogo de Aprobación de Épicas definen ya las actividades necesarias para realizar la gestión centralizada de todas las iniciativas de la empresa desde que son ideas en estudio hasta que se convierten en programas o proyectos aprobados para su ejecución. Normalmente un proyecto inicia con la venta hacia un cliente, sin embargo, cuando se requiere alguna característica del producto debido a la evaluación de los objetivos del mismo, se puede crear un requerimiento interno para la mejora del producto en base a la evaluación de la gerencia con el Product Manager.

2. **Análisis Inicial**: Este subproceso servirá para definir cómo, con quién, versión de producto que debe usarse y en qué se va a trabajar como parte del producto de software, es decir se define el grupo de trabajo (célula ágil), se definen las épicas del producto y por último se evalúa que tecnología es la más óptima para cumplir tanto con los requerimientos del cliente, como con la visión de la empresa explicada anteriormente. Este subproceso lo llevan a cabo el product Manager, Product Owner y el System Architect que se designen al proyecto, mediante una reunión de análisis de proyecto.

3. **Definición de Características**: En esta fase, se definirán a detalle las características del producto de software en base a los requerimientos del análisis preliminar. Estas características son definidas por el Product Owner, Product Manager, y con el apoyo técnico del System Architect. El resultado será una pila de características o tareas. Además, al definir estas características se hace participe al cliente para que ningún requerimiento quede fuere del alcance del proyecto y se tomen las debidas acciones con sus deseables.

4. **Estimación de Características**: En este subproceso, se estimará el tiempo en días de desarrollo que toma el realizar cada una de las características del producto de software. Este trabajo será realizado por el Product Owner, Product Manager y System Architect en conjunto, para llegar a un consenso en lo que se define como puntos de esfuerzo en días.

5. **Priorización de Riesgos**: En este subproceso, se evalúa y prioriza las principales características del software de acuerdo al algoritmo Weighted Shortest Job First o WSFJ. Este es un algoritmo de programación y priorización de trabajo para un ambiente donde la duración de los componentes de trabajo varía, así como también el coste de la demora. La fórmula de cálculo es la siguiente:

   WSJF = (Valor de Negocio/Usuario + Valor del Tiempo + Reducción de Riesgo & Valor de Oportunidad) / Tamaño del trabajo

   Dónde:

- Valor de negocio o usuario es el valor que la da la empresa a la característica o feature. Este valor se lo podría dar en un rango de 0 a 10.

- El valor del tiempo es la medición de un plan b por si algo sale mal en el proyecto. Este valor se lo podría dar en un rango de 0 a 10.

- Reducción de Riesgo & Valor de Oportunidad es donde se asigna un valor a la necesidad de la empresa en cuestión para disminuir riesgos o la generación de oportunidades de negocio que se derivan de la actual característica en el producto.

- Tamaño del trabajo es donde se le asigna un estimado de tiempo a una característica del producto o servicio ya puesto en producción.

Este trabajo es realizado por el Product Owner, Product Manager, y System architect. El resultado o salida será el documento de pila de características priorizadas.

6. **Planificación del Entregable**: En este subproceso, el Product Manager, Product Owner, System Architect y Cliente, realizan una planificación inicial de entrega de las características de software, tomando en cuenta la priorización que se hizo en el paso anterior, conjuntamente con las necesidades del cliente en cuanto a la priorización. Se realiza una reunión donde se consensua una planificación de entrega viable para ambas partes.

7. **Definición de Historias**: En este subproceso, el Product Owner define un conjunto de historias de usuario descomponiendo cada una de las características definidas en los pasos anteriores. Una historia de usuario, es representar los requerimientos del software en lenguaje común con el objetivo de que se entienda claramente lo que se debe hacer. Cabe recalcar, que cada historia de usuario contendrá detalladamente los criterios de aceptación con los que puede ser entregada. Estas historias son escritas a detalle por el Product Owner en base las características, para más tarde ser validadas por el Product Manager y el System Architect en cuanto a criterios de aceptación, funcionalidad, esfuerzo requerido y requerimientos técnicos. Por otro lado, en este subproceso se definirán claramente las dependencias con otros módulos que puedan afectar el desarrollo del producto.

8. **Estimación de Historias**: En este subproceso, el Product Owner define tiempo y esfuerzo inicial que se necesitarán para realizar un conjunto de historias de usuario, tomando en cuenta la opinión del Product Manager y System Architect.

9. **Priorización por Valor y Riesgo**: En esta subproceso, el Product Owner y System Architect evaluarán el riesgo técnico que existe para la realización de una historia de usuario, y tomarán en cuenta las dependencias que se tienen con otros módulos y que fueron identificadas en la definición de historias.

10. **Planificación del Sprint**: En este subproceso propio de la metodología SCRUM, el equipo de desarrollo de software o célula ágil conjuntamente con

el Scrum Master, estiman tiempos para las historias de usuario definidas anteriormente, con la finalidad de que puedan ser entregadas en un lapso de dos semanas y que al mismo tiempo busquen cumplir con los criterios de aceptación dados. En este paso se evalúa la capacidad en horas de trabajo que tiene el equipo para poder trabajar. Para poder planificar el tiempo de desarrollo de las historias, se procede a dividir a cada una de ellas en tareas, y el equipo de desarrollo se compromete a entregarlas al final del sprint. Dentro de esta planificación de Sprint, el Planning Poker es una técnica para que los miembros del equipo opinen al estimar el esfuerzo que se necesita al desarrollar cada una de las tareas de las historias de usuario. Se le denomina Planning Poker porque hace uso de cartas numeradas, que normalmente tienen los números de la serie de Fibonacci hasta el número 13, es decir 1,2,3,5,8,13. Además, agregamos las cartas de incógnita e infinito. La carta de incógnita se la debe asignar cuando no se conoce el esfuerzo que implica realizar una tarea, y la carta de infinito para las tareas que son demasiados extensas y deben ser divididas en tareas más cortas. Generalmente los números representan una unidad de tiempo, y normalmente son horas. El motivo por el cual se usa la serie de Fibonacci en las cartas del planning poker es porque mientras más grande es el número que se le da a una historia, mayor es la probabilidad de errar en la estimación. Es decir, lo que se busca en el planning poker es acertar en la estimación mediante la asignación de números bajos a las historias de usuario y el compromiso de cada uno de los miembros del equipo ágil.

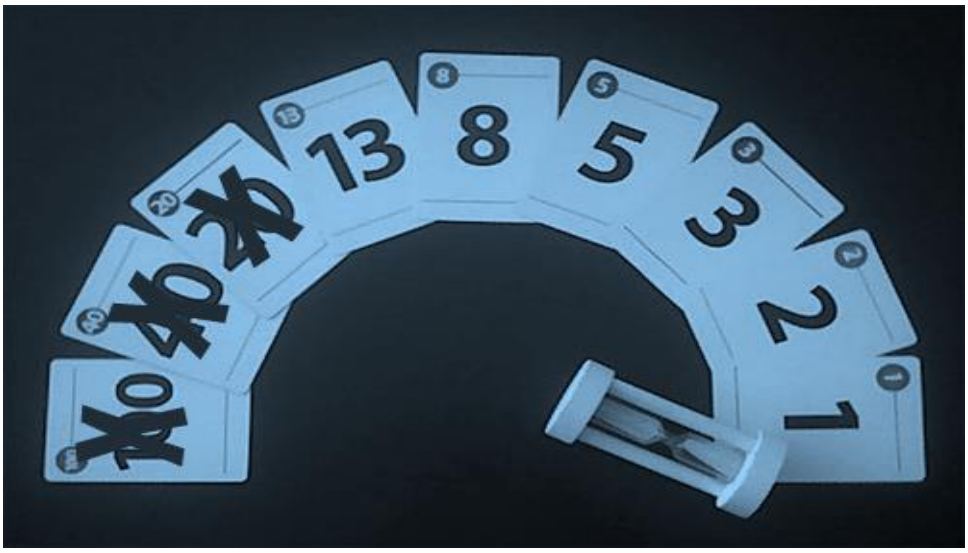

Figura 15 Juego de Cartas de Planning Poker
Fuente. Empresa de Desarrollo de Software analizada, 2016

Los pasos para poder realizar un planning poker correctamente son:

- Juntar al equipo en una mesa de reunión: En la mesa de reunión se les da a cada miembro del equipo ágil un juego de cartas de Planning Poker con la serie de Fibonacci desde el 1 al 13.

- El Product Owner o Dueño del Producto lee la historia de usuario: El Product Owner empieza por dar a conocer una historia de usuario, en caso de que no todos los integrantes del equipo ágil entiendan la historia, el Product Owner deberá despejar las dudas e incógnitas que nazcan de la misma.

- Votación de los miembros del equipo: Cada uno de los integrantes del equipo ágil deberá votar al mismo tiempo, asignando un valor de unidad que tomará en esfuerzo resolverla. Cuando se ha votado se pueden presentar dos opciones que son: buscar la unanimidad al hacer que nos expliquen las votaciones que más difieren entre sí hasta llegar a un acuerdo, o la segunda opción, que buscará

rechazar las estimaciones máximas y mínimas, hasta quedarnos con la estimación media más repetida.

- Repetir el proceso para cada historia de usuario: Debido a que tendremos varias historias de usuario con las metodologías ágiles, tendremos que repetir la votación para cada una de ellas tomando en cuenta que la iteración tiene un límite de tiempo y deberían encajar en ese lapso de tiempo.

  El resultado del Planning Poker será complementado con el tablero Kanban, al asignar a cada historia de usuario una unidad que representa el esfuerzo en resolverla y nos da una visión general sobre todo el esfuerzo que implica toda la iteración o sprint. Cada columna del tablero Kanban simbolizará un estado específico de flujo de tareas, y nos servirá para identificar el actual estado del proyecto a desarrollarse. Conforme las tareas vayan avanzando por el flujo de trabajo, se las irá cambiando de estado hasta llegar al estado de tarea finalizada. Mientras se desarrollen las tareas en los siguientes subprocesos se irán cambiando de estado tanto las tareas como las historias que las contienen.

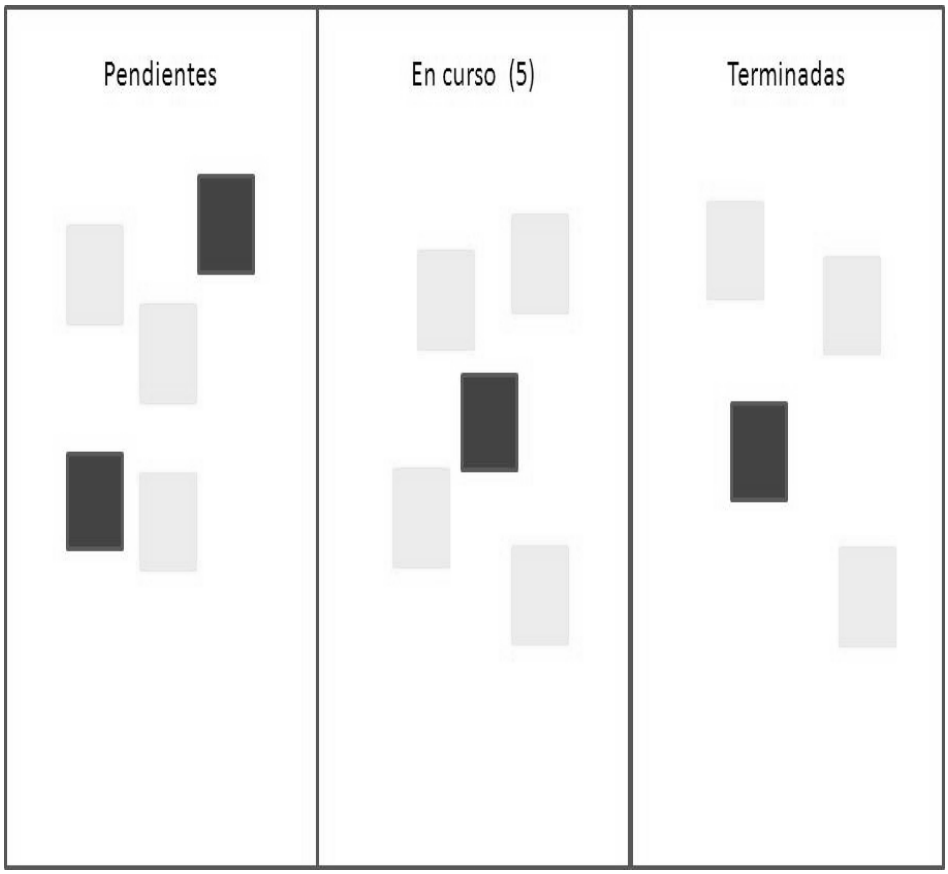

Figura 16 Tablero Kanban
Fuente. Empresa de Desarrollo de Software analizada, 2016

11. **Desarrollo y Pruebas**: En este subproceso, el equipo de desarrollo de software o célula ágil programan y resuelven las historias de usuario descompuestas en tareas con las correspondientes pruebas del Tester. Las persona que valida que una historia ha sido finalizada completamente es el Product Owner seguido del Product Manager. El desarrollador de software del equipo ágil se compromete a informar el estado de cada tarea mediante el uso de un tablero Kanban con estados como: por hacer, en progreso, terminada, en pruebas y aprobada. Para que una historia pase a estado de pruebas y el tester cumpla su responsabilidad, el desarrollador debe cumplir todos los criterios de aceptación de la historia y a su vez resolver todas las

tareas que componen la historia. Una vez, el tester haya aprobado las historias se debe enviar para la revisión e inspección de las mismas al Product Owner y Product Manager.

12. **Revisión del Sprint y Retrospectiva**: En este subproceso propio de la metodología SCRUM, el equipo de desarrollo de software o célula ágil conjuntamente con el Product Owner, revisan las historias que han sido finalizadas y evalúan las historias que no se lograron cumplir. Muchas veces ocurren problemas inesperados de programación o dependencia con otros módulos que no permiten finalizar las historias a plenitud. Por otro lado, la célula ágil y el Scrum Master realizan un proceso de retroalimentación mediante un documento de retrospectiva, donde se detallan los principales problemas tenidos en el sprint y como se deberían resolver los mismos.

13. **Revisión, Inspección y Adaptación**: En este subproceso, el equipo de desarrollo de software o célula ágil, envía a instalar las historias finalizadas en el sprint en un ambiente controlado por la empresa. Para poder realizar dicha instalación se usan recursos del equipo de administración de la configuración y principalmente se buscan errores en los instaladores, para que estos no se repitan en un ambiente del cliente. En caso de encontrar algún error en los instaladores, el equipo de desarrollo deberá corregir el mismo y se planificará un tiempo de corrección para el siguiente sprint.

14. **Liberación del Entregable**: En este subproceso, el equipo de administración de la configuración se encarga de hacer la instalación de las funcionalidades a

los ambientes de pruebas del cliente. Cabe recalcar, que los releases son planificados para entregarse solo cuando se hayan cumplido cierto número de historias de usuario que componen ciertas funcionalidades completamente.

15. **Pruebas de Aceptación del Usuario**: En esta subproceso, equipos de pruebas del cliente revisan la funcionalidad a detalle y de encontrarse algún error, ellos envían mediante una herramienta de software un bug que tendrá que ser resuelto inmediatamente. Estos bugs son planificados en los Sprints de desarrollo, dando como resultado que la célula ágil atienda nueva funcionalidad, así como también errores reportados por el cliente.

16. **Producción**: En esta subproceso, se envía a un líder técnico de la célula ágil conjuntamente con el Product Owner, a instalar el software entregado. Esta etapa se la realizará una vez se haya finalizado todo el software, y se planifica tiempos con fechas de menos impacto para el cliente. Este subproceso concluye el proceso de desarrollo de software del módulo de Banca en Línea

17. **Operación**: En este subproceso, el cliente tendrá un soporte por parte de la célula ágil en caso de que se diera un error en producción y se necesite su inmediata solución. Debido a la metodología y SCRUM y el marco de trabajo SAFe, no es común que se den errores de este tipo.

3.6.1 Procesos de Apoyo

Para poder finalizar a plenitud el proceso de desarrollo de software del módulo de Banca en Línea existen procesos de apoyo que ayudan a lograrlo, y se describen a continuación:

- **Parametrización:** Este proceso establece planes, estrategias y procedimientos con criterios de aceptación objetivos, que son utilizados para realizar una parametrización de ambientes exitosa. La parametrización se entiende como toda la información inicial necesaria que debe tener la base de datos, para el correcto funcionamiento del software entregado.

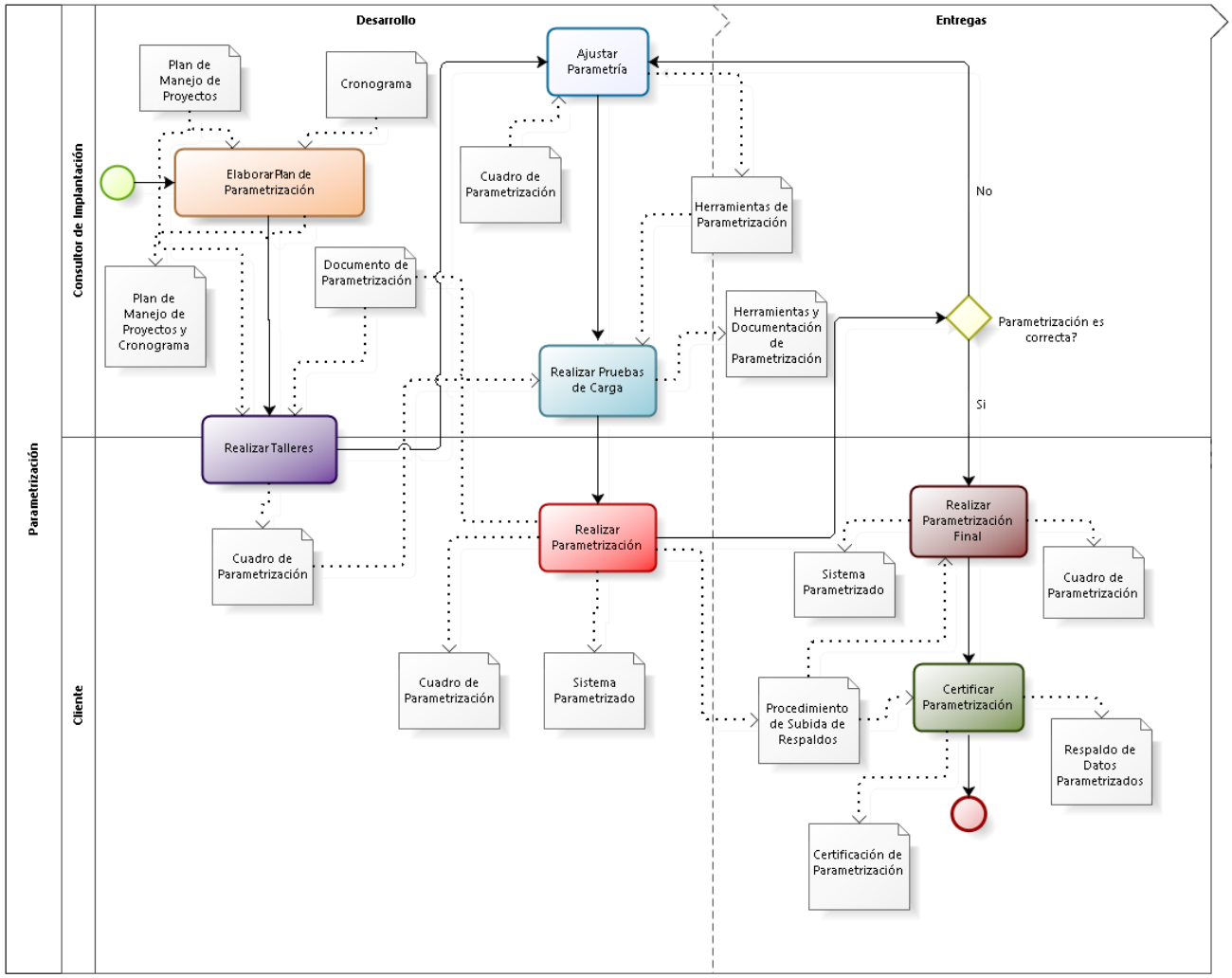

Figura 17 Proceso de Apoyo Parametrización
Fuente. Daniel Jarrín, 2016

- **Capacitación:** Este proceso permite transferir el conocimiento técnico y administrativo del software para que los clientes puedan utilizar todas las funcionalidades diseñadas en el mismo.

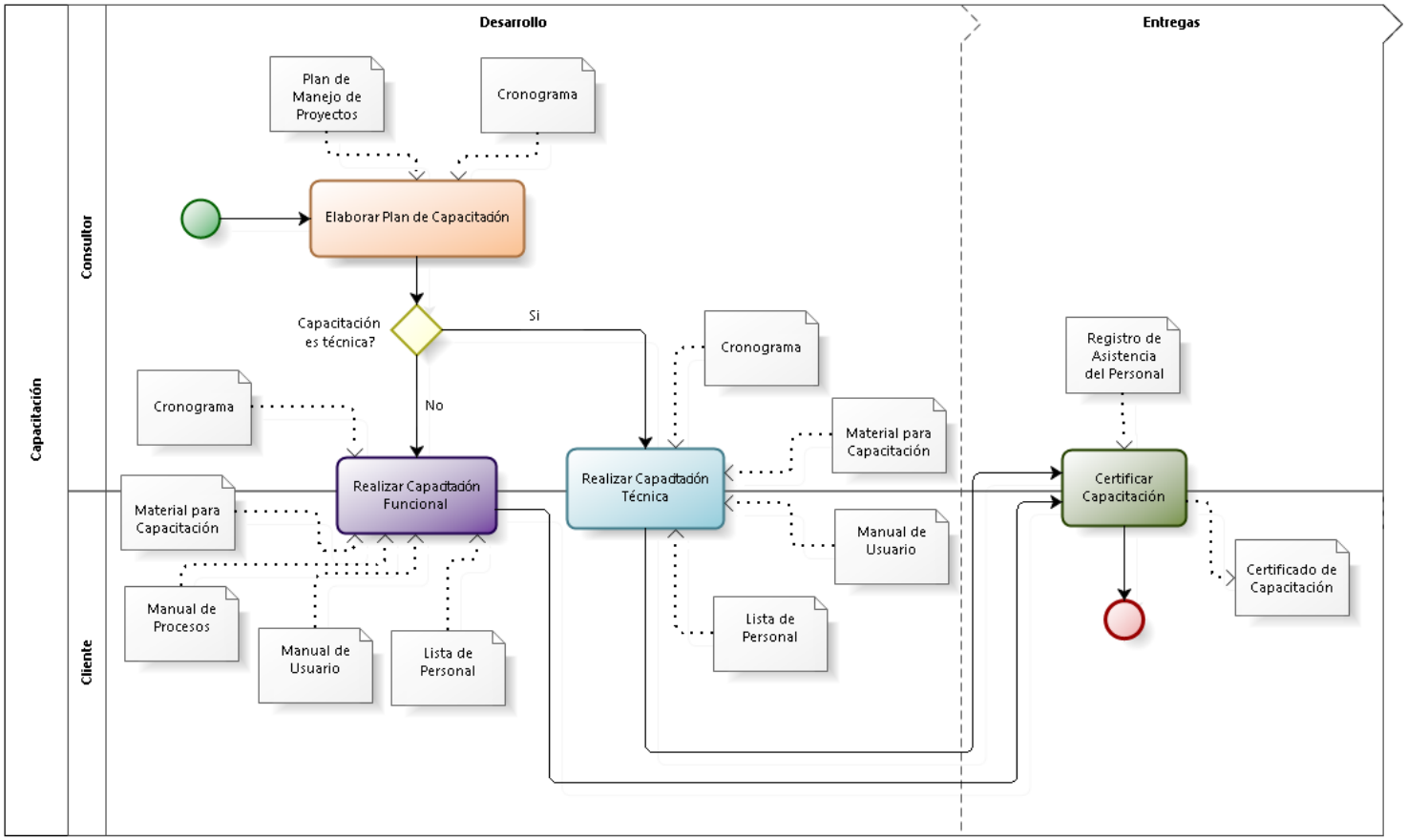

Figura 18 Proceso de Apoyo Capacitación
Fuente. Empresa Desarrolladora de Software, 2016

- **Administración de la Configuración:** Este proceso permite establecer mecanismos para gestionar ambientes, versiones del software y mantener su integridad durante y posterior a la ejecución del proyecto, así como catalogar y preparar los entregables para el cliente.

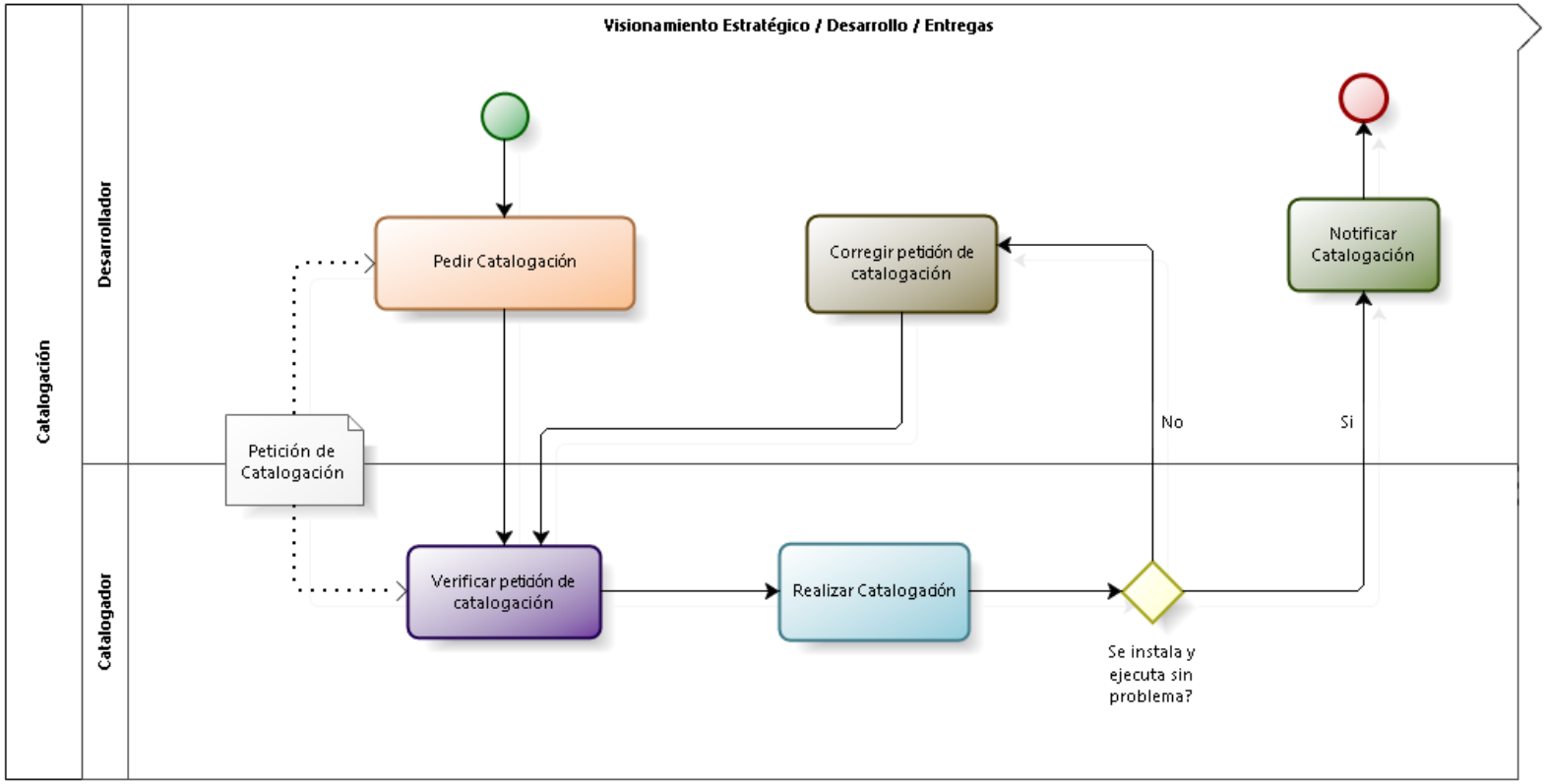

Figura 19 Proceso de Apoyo Catalogación

Fuente. Daniel Jarrín, 2016

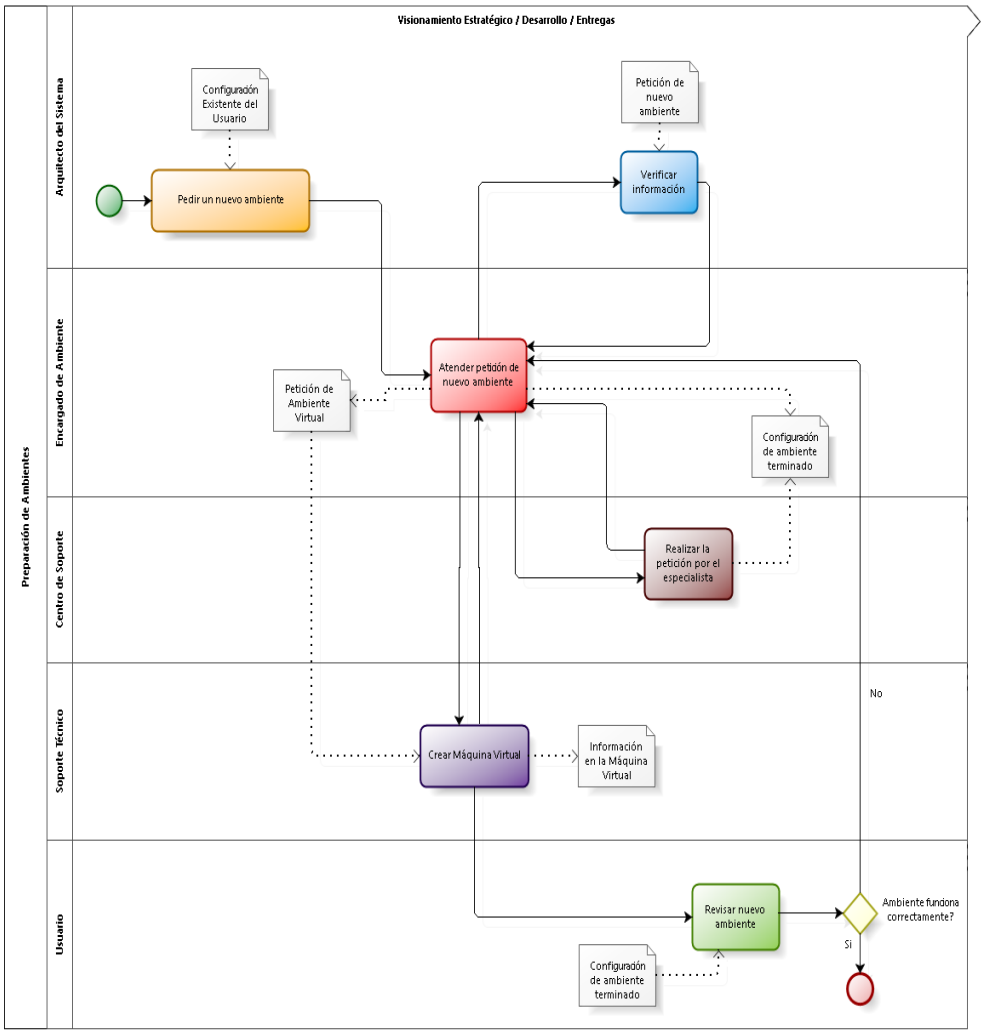

Figura 20 Proceso de Apoyo Preparación de Ambientes
Fuente. Daniel Jarrín, 2016

- **Gestión de Proyectos:** Este proceso es el nivel de decisión y gestión más alto del proyecto, y coordina a todos los procesos de ingeniería y a todos los procesos de apoyo involucrados en un proyecto, este proceso permite gestionar los objetivos del proyecto y los riesgos del mismo para entregar un producto de calidad al cliente, cuidando las variables principales del proyecto: costos, tiempos, calidad y alcance. Se ejecuta cuando un proyecto ha sido aprobada en el comité de Portafolio de la empresa, y por lo tanto se ha asignado un presupuesto,

un Gerente de Proyecto y recursos necesarios para que se convierta en un proyecto de desarrollo.

3.7 ÁREAS RELACIONADAS AL PROCESO

Las áreas que se involucran directamente con el proceso de desarrollo de software, y que son encargados de cumplir sus procedimientos son los siguientes:

- **Soporte de Administración y Configuración:** Esta área se encarga de administrar los procesos para gestionar ambientes, versiones del software y mantener su integridad durante y posterior a la ejecución del proyecto, así como catalogar y preparar los entregables para el cliente. Actualmente está conformada de 10 personas que se coordinan en distintos horarios para poder cumplir con demandas del cliente externo e interno.

- **Administración de Proyectos:** Esta área se encarga de designar equipos ágiles de trabajo que permitan cumplir proyectos en costo y tiempo. Actualmente tienen continua interacción con los especialistas de cada módulo de la empresa, para poder estimar el esfuerzo que implica desarrollar diferentes proyectos. Además, dan seguimiento sobre el avance del proyecto para evitar que este se salga de presupuesto o afronte un retraso.

3.8 RECOLECCIÓN DE INFORMACIÓN E IDENTIFICACION DE PUNTOS DE MEJORA DEL PROCESO

Al observar y comprobar con tiempos obtenidos de un proyecto finalizado hace 3 meses, con una duración de desarrollo de 10 meses, y donde se aplicó el proceso de desarrollo de Software en el módulo de Banca en Línea, se lograron identificar ciertas falencias descritas a continuación:

- **Planificación de historias y tareas muy optimistas:** De acuerdo a datos obtenidos en el software de monitoreo, en promedio se cumplen un 74% de la tareas e historias a las que el equipo de desarrollo de software se compromete en el Sprint. Varias de las veces estos tiempos que el equipo de desarrollo de software estima son comparados con la planificación inicial del Producto Owner y se modifican en caso de ser más grandes. Otras veces, la planificación de tareas por parte del equipo de desarrollo de software, no está exigiendo una descomposición de una historia de usuario a detalle. Las tareas que se planifiquen al momento de realizar un Sprint deberían cubrir a plenitud el desarrollo de una historia de usuario y sobre todo tomar en cuenta eventos inesperados e interacción con otros módulos de la empresa.

Tabla 5:
Porcentajes de cumplimiento por sprint de la célula ágil

Sprint	% Cumplimiento
Sprint 1	80%
Sprint 2	68%

Continúa

Sprint	% Cumplimiento
Sprint 3	80%
Sprint 4	60%
Sprint 5	60%
Sprint 6	63%
Sprint 7	76%
Sprint 8	72%
Sprint 9	76%
Sprint 10	84%
Sprint 11	84%
Sprint 12	84%
Sprint 13	76%
Sprint 14	80%
Sprint 15	84%
Sprint 16	72%
Sprint 17	76%
Sprint 18	68%
Sprint 19	72%
Promedio	**74%**

- **Deuda técnica alta del equipo:** De acuerdo a datos obtenidos en el software de monitoreo, en promedio se tuvo una deuda técnica del 68%. Se entiende por deuda técnica la falta de pruebas automatizadas y la modificación de código que posiblemente no cumplía a cabalidad los estándares de programación de la empresa.

Tabla 6:
Porcentajes de Deuda Técnica del equipo ágil

Sprint	% Deuda Técnica
Sprint 1	40%
Sprint 2	56%
Sprint 3	68%
Sprint 4	60%
Sprint 5	60%
Sprint 6	67%
Sprint 7	76%
Sprint 8	76%
Sprint 9	72%
Sprint 10	76%
Sprint 11	80%
Sprint 12	77%
Sprint 13	60%
Sprint 14	60%
Sprint 15	67%
Sprint 16	80%
Sprint 17	80%
Sprint 18	73%
Sprint 19	60%
Promedio	**68%**

- **Parametrización faltante o incorrecta en ambientes de la empresa:** De las reuniones de retrospectiva se registraron 4 veces problemas de parametrización que no permitían entregar o culminar correctamente varias historias de usuario durante el desarrollo del proyecto. Esto ocasionó una pérdida de 48 horas a los 3 integrantes de la célula ágil.

- **Versión de Producto base no controlada al empezar el proyecto:** Debido a que el proyecto consistía en incrementar una funcionalidad a un proyecto previamente puesto en producción, se tuvo la dificultad para poder empezar el desarrollo ya que el equipo de administración de la configuración no tenía debidamente custodiada la versión anterior y se tuvo que recuperar algunas versiones de los fuentes, ocasionando un retraso de 1 semana para todo el equipo.

- **Preparación de ambientes toma más tiempo del debido:** De acuerdo al histórico del proyecto, la preparación de ambientes para la instalación inicial del primer sprint tomó alrededor de 56 horas hombre, valor que es 4 veces más del comúnmente utilizado para lo antes mencionado. Este trabajo es realizado por el equipo de administración de la configuración.

- **Uso de herramientas de trabajo que incrementen eficiencia en el trabajo del desarrollador:** Actualmente se está buscando con los System Architects definir un listado de herramientas oficiales que nos permitan ser más eficientes y productivos a la hora de realizar un desarrollo o dar seguimiento a un error. Lo complicado de esta tarea radica en que las herramientas que se escojan deben poder aplicarse en toda la empresa porque normalmente tienen un costo y no justifica el hacerlo por cada área, módulo o submódulo.

- **Tiempos de capacitación no planificada:** Debido a la necesidad de cumplir tiempos planificados o por rotación del personal, el grupo de desarrolladores de

software fue cambiando en el tiempo, específicamente se agregaron 3 integrantes nuevos al proyecto. La capacitación técnica que se les debe dar a los nuevos integrantes para que empiecen a aportar con su conocimiento, varias de las veces no fue tomado en cuenta en la planificación del Sprint por la premura del tiempo, ocasionando la reducción del tiempo de desarrollo de las personas que la daban.

3.8.1 Análisis de los Puntos de Mejora

Al analizar las principales causas de los problemas encontrados en el proceso mediante el uso de diagramas de causa-efecto y conjuntamente con todo el equipo ágil, se identificó lo siguiente:

- **Planificación incorrecta de tareas e historias de usuario:** Estas son las causas identificadas en el proceso, para el punto de mejora de planificación de historias y tareas muy optimistas.

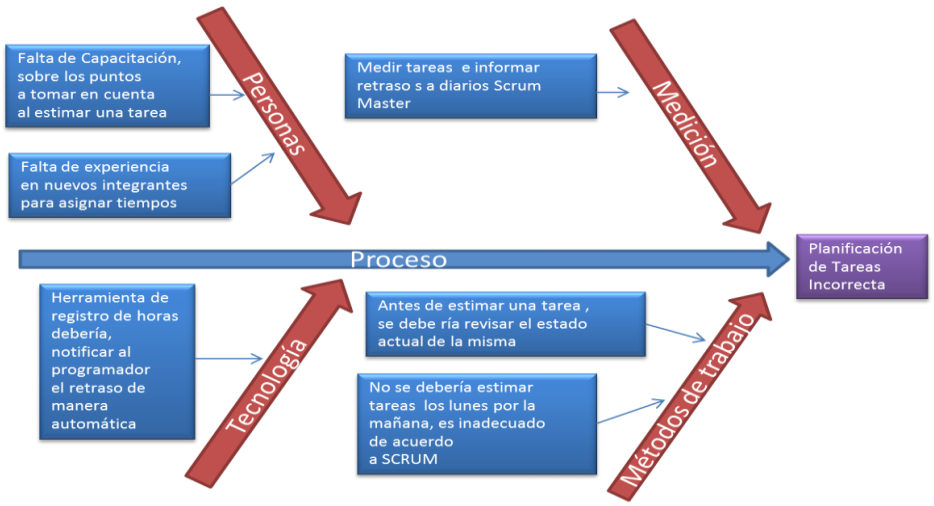

Figura 21 Diagrama Causa Efecto para la correcta Planificación de Tareas
Fuente. Equipo ágil de Empresa Desarrolladora de Software analizada, 2016

- **Deuda Técnica alta del equipo de Desarrollo:** Estas son las causas identificadas para la mejora de este problema.

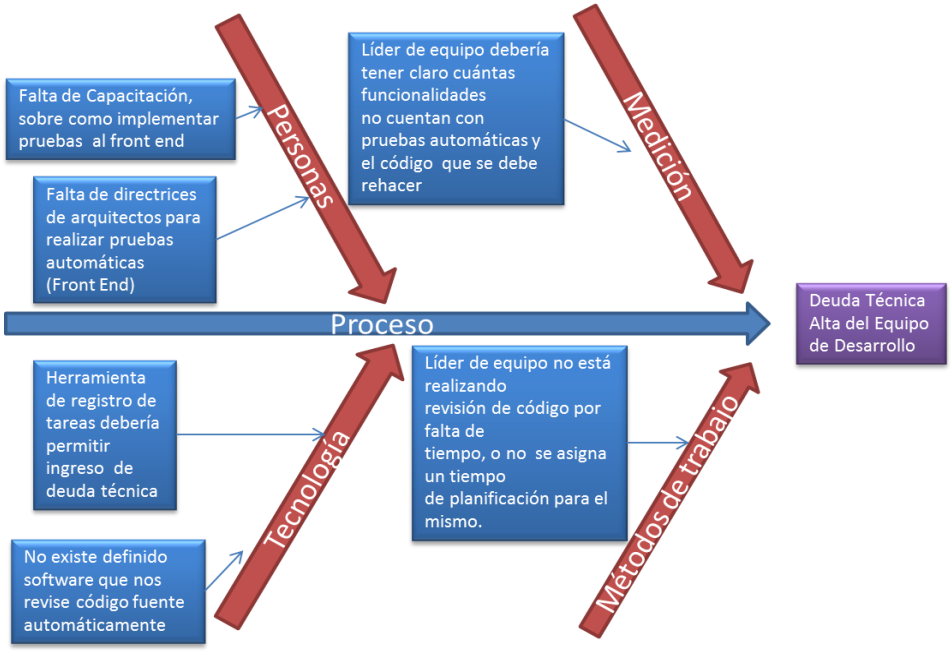

Figura 22 Diagrama Causa Efecto sobre Deuda Técnica de equipo alta
Fuente. Equipo ágil de Empresa Desarrolladora de Software analizada, 2016

- **Versión de producto base no controlada en cuanto a fuentes:** Estas son las causas identificadas para la mejora de este problema.

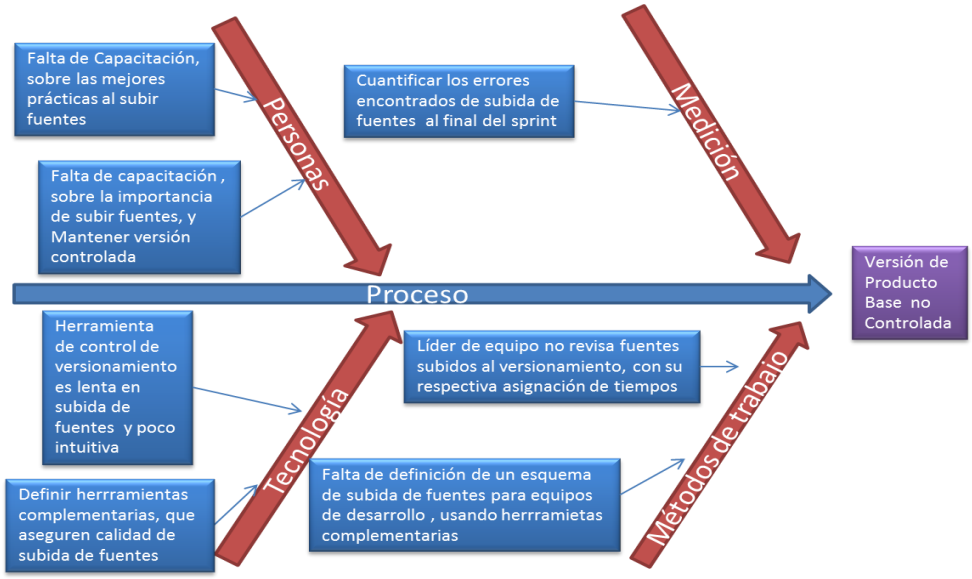

Figura 23 Diagrama Causa Efecto para control de versión de producto Base
Fuente. Equipo ágil de Empresa Desarrolladora de Software analizada, 2016

- **Parametrización faltante o incorrecta en ambientes de la empresa:** Estas son las causas identificadas para la mejora de este problema:

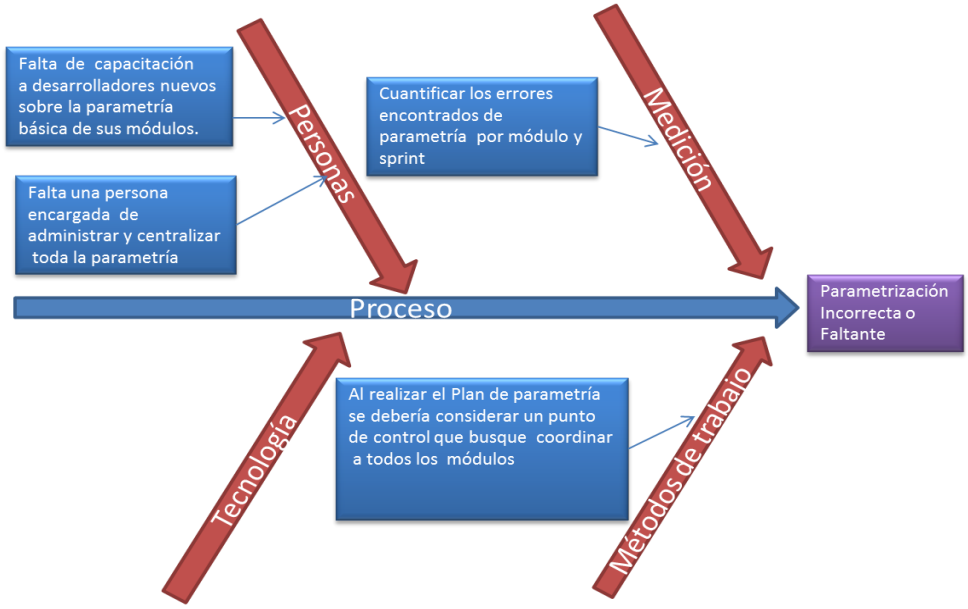

Figura 24 Diagrama Causa Efecto para Parametrización Incorrecta o Faltante
Fuente. Equipo ágil de Empresa Desarrolladora de Software analizada, 2016

- **Preparación de ambientes toma más tiempo del debido y daños:** Estas son las causas identificadas para la mejora de este problema.

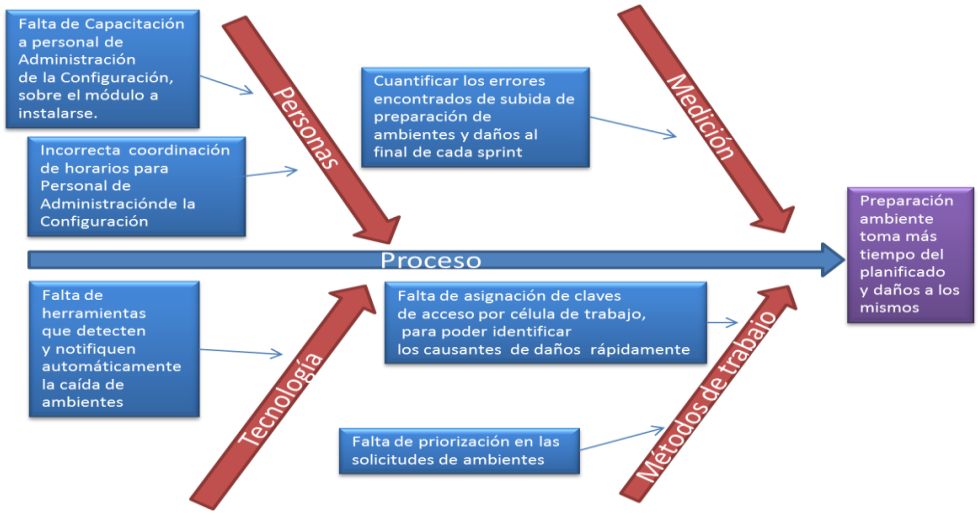

Figura 25 Diagrama Causa Efecto para Preparación de Ambientes Lenta y Daños
Fuente. Equipo ágil de Empresa Desarrolladora de Software analizada, 2016

- **Uso de herramientas de trabajo que incrementen eficiencia en el trabajo del desarrollador:** Estas son las causas identificadas del problema.

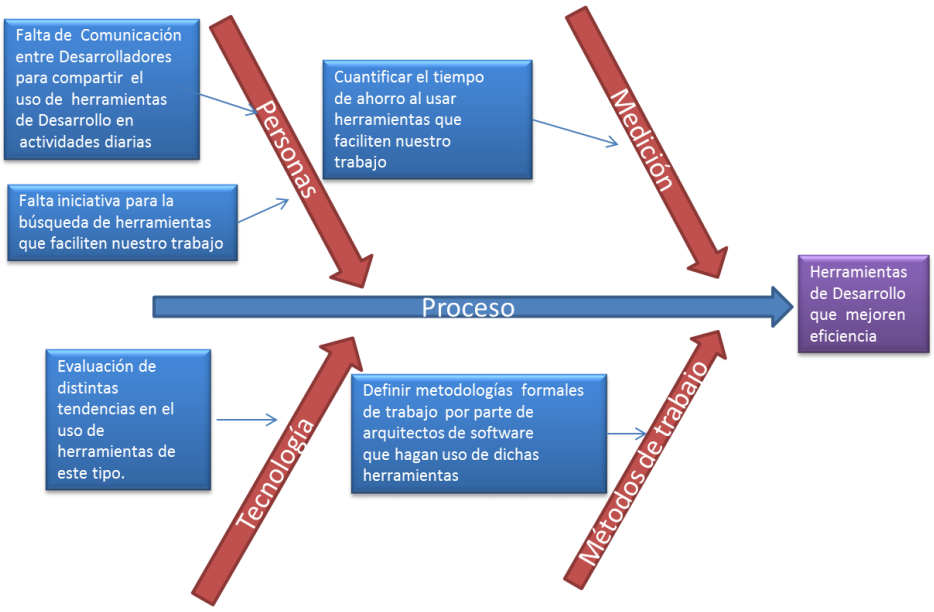

Figura 26 Diagrama Causa Efecto para uso de herramientas que mejoren eficiencia
Fuente. Equipo ágil de Empresa Desarrolladora de Software analizada, 2016

Necesario mencionar, que las causas analizadas nacieron de la utilización de la herramienta denominada lluvia de ideas dentro de las reuniones de retrospectiva. Estos problemas con sus principales causas, ocasionan retrasos dentro del proceso de desarrollo de software en el módulo de banca en Línea de la empresa analizada.

3.9 ANÁLISIS DE MADUREZ DEL PROCESO DE DESARROLLO DE SOFTWARE

Una vez definidos los problemas y causas del proceso de desarrollo de software del módulo de Banca en Línea, continuaremos a realizar el análisis de madurez del

mismo, para lo cual usaremos el Modelo de Madurez de Capacidades o CMM. Este modelo tiene como objetivo el definir los elementos para llevar a buen término a los procesos de desarrollo de software. Además, provee varios procedimientos enfocados en evaluar y corregir los procesos de desarrollo, mantenimiento e implementación de software. El modelo CMM maneja los siguientes niveles:

1. **Inicial:** Este nivel es el más básico y aquí las organizaciones no tienen espacios favorables para el desarrollo de software, a pesar de que el trabajo se lo hace técnicamente, y con un personal dispuesto a esforzarse, pero sin planificar ni estrategias que dan como resultado, retrasos e incrementan los costos previstos.

2. **Repetible:** En este nivel estarían las organizaciones que tienen implementado una planificación, acompañamiento del proyecto, métricas o indicadores básicos y la gestión del proceso, sin embargo, hay el riesgo de no llegar a las metas propuestas.

3. **Definido**: En este nivel se ha conquistado una madurez, llegando a estándares, métricas o indicadores más detallados, métodos o procedimientos para coordinación entre grupos o áreas de trabajo. Además, se suele implementar técnicas de revisión de pares.

4. **Gestionado**: En este nivel se puede identificar a organizaciones donde se tienen normas importantes de calidad y un conjunto de indicadores o métricas

significativas que se las emplea constantemente para tomar decisiones y en la gestión de riesgos. En este nivel se dispone de un software de muy buena calidad.

5. **Optimizado**: En este nivel, la empresa busca la mejora continua constantemente. Toda la empresa está centrada en mejorar sistemáticamente los procesos a través del empleo de los indicadores e innovando el proceso.

3.9.1 Áreas de Procesos Claves KPA's

El Modelo de Madurez de Capacidad define un conjunto de áreas de proceso como claves para poder tener un mejoramiento en los procesos de desarrollos de software, y se los conoce como KPA's. El nivel de madurez de una empresa vendrá dado si y solo sí todas las KPA's del nivel han sido cumplidas. Es importante resaltar que el único nivel que no posee KPA's es el nivel inicial o por su numeración 1.

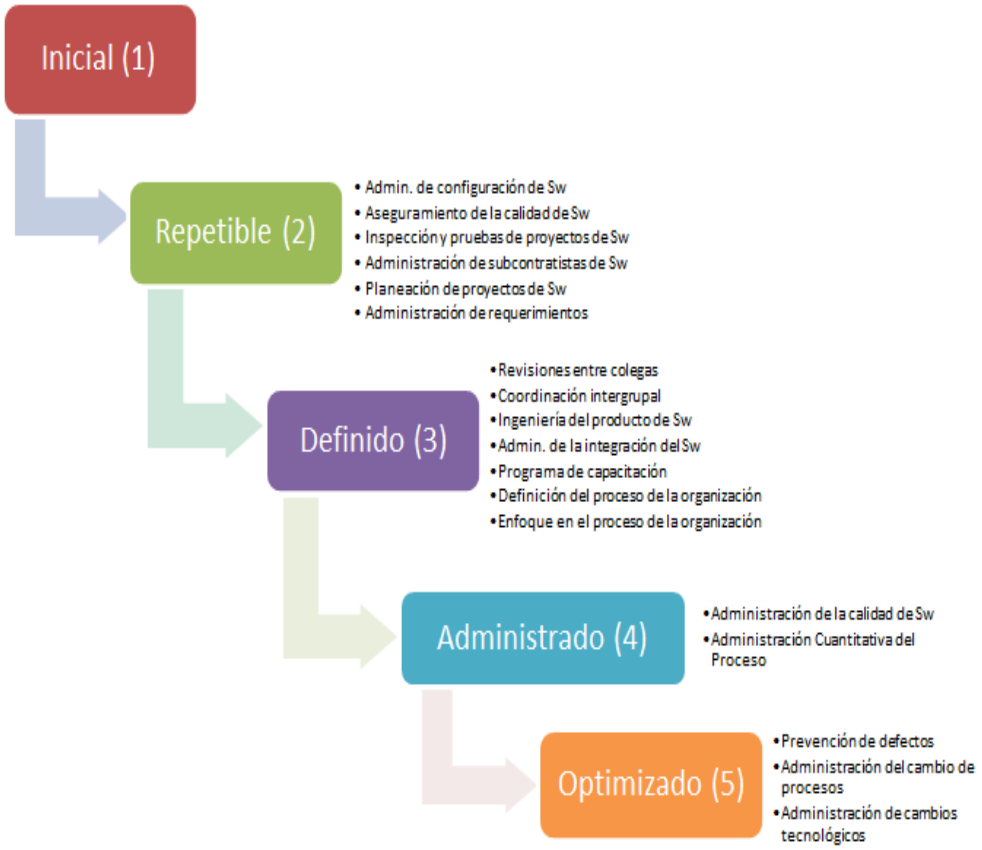

Figura 27 KPA's del modelo CMM
Fuente. Daniel Jarrín, 2017

Dentro del Modelo de Madurez de Capacidad CMM cada uno de los KPA's se dividen o distribuyen de acuerdo a tres categorías del proceso y son: gerencia, organizacional e ingeniería. El cuadro que se muestra a continuación mostrará la distribución a detalle.

Tabla 7:
Distribución de KPA's por áreas de proceso

Categorías de Proceso por nivel	Gerencia (Planeación del proyecto de software, administración, etc.)	Organizacional (Capacitación, infraestructura, etc.)	Ingeniería (Análisis de requerimientos, diseño, codificación, pruebas, etc.)
5		Administración de cambios tecnológicos	
	Administración del cambio de procesos		Prevención de defectos

Continúa

Categorías de Proceso por nivel	**Gerencia (Planeación del proyecto de software, administración, etc.)**	**Organizacional (Capacitación, infraestructura, etc.)**	**Ingeniería (Análisis de requerimientos, diseño, codificación, pruebas, etc.)**
4	Administración cuantitativa del proceso	Administración de la calidad de software	4
3	-Administración de la integración del software -Coordinación intergrupal	- *Enfoque en el proceso de la organización* - Definición del proceso de la organización -Programa de capacitación	*-Ingeniería del producto de software* -Revisión entre colegas
2	-Admin. de Requerimientos -Planeación de proyectos de software -Seguimiento y supervisión de proyectos de software -Administración de subcontratistas de software -Aseguramiento de la calidad de software -Administración de la configuración de software		
1	Procesos Ad-hoc		

3.9.2 Resultado de madurez de la empresa

De acuerdo, a lo expuesto en los distintos niveles del modelo de Madurez de Capacidades CMM y con la evaluación de los KPA's expuestos anteriormente, la empresa de desarrollo de software del estudio se ubicaría en el nivel 3 de la misma, debido a que cumple todos los KPA's del nivel 2 y 3, como son:

- Se dan revisiones entre integrantes del grupo porque la metodología SCRUM permite revisión entre pares.

- Posee una coordinación intergrupal debido a que SCRUM lo hace intrínsecamente.

- Se tiene una ingeniería de software sobre el producto, en todos los niveles del desarrollo.

- Se tiene una administración de la integración de software entre áreas y se considera la integración con herramientas del cliente.

- Se tiene un programa de capacitación interno, sin embargo se ha identificado que puede ser mejorado.

- Se tiene un proceso de desarrollo de software que afecta a toda la organización.

- Dentro de la empresa se maneja un enfoque en procesos alrededor de todas las áreas.

- Se tiene un proceso de administración de la configuración que afecta a todas las áreas de la empresa.

- Se tiene un mecanismo de aseguramiento de la calidad, dado por los procesos y las metodologías sobre la cual se basan sus procesos, como lo es SCRUM.

- Se tiene una inspección y pruebas de proyectos de software dado por la metodología SCRUM.

- Existe una planificación dentro de los proyectos de software que es dada por el Sprint Planning de la metodología SCRUM.

- Se administran los requerimientos del cliente al usar el marco de trabajo Scaled Agile Framework SAFe. SAFe permite participar al cliente en la definición de las características del software.

Además de los KPA's antes nombrados, la empresa tiene varios indicadores sobre el proceso de desarrollo de software como por ejemplo: Velocidad de Equipo en puntos, Porcentaje Cumplimiento Sprint, Valor de Negocio entregado al cliente en puntos, Porcentaje de Deuda Técnica, Costo unitario de Producción, Productividad por Sprint, Porcentaje de cumplimiento de Trabajo Objetivo o Meta, Eficiencia por sprint, Costo por Horas Hombre en Sprint, y el número de errores cometidos durante el proceso. Por otro lado, el proceso podría subir de nivel al considerar mejores métricas que abarquen todos los problemas del proceso como por ejemplo los números de errores por tipo, conjuntamente con el tiempo perdido por cada uno. Aquí, el área de gestión de proyectos debería trabajar más de cerca para tomar decisiones más ágilmente con respecto a las métricas que se tienen en la empresa.

4. PROPUESTA DE MEJORA AL PROCESO DE DESARROLLO DE SOFTWARE EN EL MÓDULO DE BANCA EN LÍNEA

4.1 DETALLE DE MEJORAS AL PROCESO

De acuerdo a los puntos identificados de mejora del proceso y sus principales causas, mi propuesta se basa en los siguientes puntos:

4.1.1 Mejora en la planificación de tiempos de historias y tareas

Se propone que el Product Owner informe en una reunión rápida a los integrantes del equipo las posibles tareas a realizarse en el siguiente sprint y sus posibles encargados, con el objetivo que durante 4 horas del día lunes, se revisen el estado actual de las tareas y cada uno pueda empaparse mejor de la realidad de la misma y estimar de mejor manera los tiempos los días martes que serían de inicio de Sprint. Dentro de la revisión cada miembro del equipo debe identificar cada una de las tareas a realizarse y los miembros junior del equipo deben apoyarse en los desarrolladores con más experiencia. Actualmente nuestras estimaciones tienen cierto grado de incertidumbre porque muchos miembros del equipo no conocen el estado de lo que van a desarrollar o modificar y sus estimaciones omiten varios puntos relacionados con la tarea.

De acuerdo con (Hundermark, 2011) la planificación de sprints no se las debe realizar los días lunes a la mañana. El equipo no se encuentra en un estado pleno de concentración, además estadísticamente es el día que se producen mayor cantidad de ausencias por enfermedad. Por lo tanto es una buena idea considerar el comienzo de sprint los días martes.

Resumiendo, media mañana del día lunes se lo puede usar para realizar pequeñas correcciones de deuda técnica que tenga el equipo del anterior Sprint, o pequeños errores encontrados, para más tarde proceder a realizar la revisión de tareas del sprint que se viene. Con esto el día martes se iniciaría la planificación y se procedería a dar la estimación de las tareas del sprint, buscando que el cumplimiento de las tareas del equipo sea siempre de un 100%. Además se buscaría que el equipo se mantenga motivado, debido a que no tuvo que trabajar horas extras para cumplir la mayoría de tareas del Sprint.

4.1.2 Mejora en la deuda técnica del equipo ágil

Se propone que durante la reunión de planificación de Sprint se asigne un tiempo al líder técnico de la célula ágil que por lo general es el System Architect, para la revisión de código de todas las funcionalidades a desarrollarse. Dicha revisión debería topar aspectos de brecha tecnológica, arquitectura, cumplimiento de estándares, mejores prácticas, y debería realizarse conforme se vayan entregando las tareas del sprint, para que en caso de encontrarse alguna deuda técnica, pueda ser corregida y entregada en ese momento.

Por otro lado, se necesita que el equipo de arquitectos de la organización, definan el estándar para la automatización de las pruebas de la capa de la vista para todos los módulos de la organización, debido a que un 50% de la deuda técnica es causada por la falta de implementación de dichas pruebas.

Además, se requiere que todos los proyectos tengan la revisión de código automático tanto a nivel de la capa transaccional como a nivel de base de datos. Actualmente se tiene definido un estándar pero se debería asignar un tiempo para que los equipos de Banca en Línea puedan ponerlo a funcionar a lo largo de todo el módulo. Este estándar ya se ha trabajado con los arquitectos de la empresa y se llegó a un consenso de uso e implementación para los diferentes módulos. El esfuerzo que se requiere para implementar esta revisión en todo el módulo de banca en línea son 40 horas.

A lo largo de todos los 20 sprints del proyecto que tomé como referencia, se reportaron 68 errores, de los cuales 20% fueron ocasionados por no tener correctamente implementadas las pruebas automáticas a nivel de la capa de vista y por la incorrecta utilización de los estándares ya definidos que posee la empresa. El tiempo usado en resolver este tipo de error fue de 192 horas en el proyecto que representa un 3% del tiempo total del desarrollo del proyecto de software. Dentro de cada sprint se asigna un promedio de 20% del tiempo de una persona del equipo ágil para la corrección de todo tipo de errores que se dan durante el desarrollo.

Tabla 8:
Errores por Sprint

Sprint	Errores
Sprint 1	2
Sprint 2	8
Sprint 3	3
Sprint 4	2
Sprint 5	22
Sprint 6	5
Sprint 7	9
Sprint 8	7
Sprint 9	2
Sprint 10	3
Sprint 11	3
Sprint 12	2
Sprint 13	0
Sprint 14	0
Sprint 15	0
Sprint 16	0
Sprint 17	0
Sprint 18	0
Sprint 19	0
Total	**68**
Ahorro por atención Errores Deuda Técnica	**192 horas en proyecto**

4.1.3 Mejoramiento en el control del producto y subida de fuentes

Debido a ser un problema muy común entre los módulos, el área de administración de la configuración conjuntamente con los arquitectos

trabajaron arduamente en las compilaciones automáticas, una vez se sube un código fuente a nuestro software de control de versión. Dichas compilaciones automáticas, nos aseguran que el código que se suba al software de control de versión compile correctamente, o caso contrario se nos notifique para que los desarrolladores revisemos si falta algún archivo de subir o algo fue subido incorrectamente. La mejora antes nombrada fue entregada hace no mucho tiempo pero se espera ya no volver a tener errores de este tipo.

Además, la metodología SCRUM es muy clara al definir una tarea como finalizada, donde esta debe estar probada, documentada, subida al software de control de versión, y validada en un ambiente de pruebas donde ya no intervienen los desarrolladores sino el área de catalogación. En caso de existir algún error el grupo de catalogadores informan al equipo, y este debe solucionarlo lo más rápido posible, para así dar por terminada dicha tarea.

Para complementar la solución de controlar de mejor manera el producto, se debería dar una capacitación formal inicial a cada miembro nuevo del equipo, cada vez que ingresen, de esta manera se podría trasmitir la importancia de subir el código fuente correctamente y evitar correcciones más tarde.

Los errores ocasionados por subir código incorrectamente deberían ser cuantificados por el SCRUM Master, en cuanto al tiempo que tomo corregirlos para de esta manera traspasar el costo al equipo. El proyecto que fue tomado como referencia, tuvo un retraso real de 40 horas para dar inicio

al desarrollo. Estos tiempos de retraso se podrían usar al final del proyecto para poder cubrir imprevistos que se dan durante el desarrollo del mismo.

4.1.4 Correcta preparación de ambientes y disminución de daños

Para lograr que el proceso de apoyo de creación de un nuevo ambiente sea eficiente se requiere que la persona que realiza la solicitud del ambiente sea un especialista encargado del producto, principalmente cuando se clonan ambientes para aumentar funcionalidades a versiones previamente entregadas. Actualmente se tiene que abrir varios incidentes a la entrega de un ambiente, porque no se tiene claro los componentes que manejaba la versión, y la persona que solicita los mismos es el SCRUM Master del equipo ágil.

Normalmente la preparación de un ambiente debería tomar entre 8 a 16 horas, sin embargo la creación de incidentes por errores encontrados a los mismos hacen que en promedio tome 56 horas, es decir 40 horas adicionales que afectan a toda una célula de 5 personas, dando como resultado 200 horas perdidas de horas hombre.

Una vez entregado el ambiente, los daños al mismo se están volviendo comunes y no existe la manera de comprobar quiénes son los causantes y cargarles el costo de arreglo del mismo. Cabe mencionar, que por los elevados costos de mantenimiento de un ambiente, estos son compartidos entre equipos del mismo proyecto, dando como resultado que un daño pueda afectar a 2 o 3 equipos al mismo tiempo. Por esta razón, se hace fundamental

la entrega de accesos de usuarios tanto para base de datos como para servidores por miembro de equipo, así cada usuario despertaría una conciencia sobre el correcto uso del ambiente y se clarificarían los costos del proyecto. Los errores antes nombrados deberían ser cuantificados por el SCRUM Master en cuánto al tiempo de solución y a cuántas personas afectaron, de esta manera se tendría claro cómo afectaron al proyecto en curso.

4.1.5 Mejoramiento en la parametrización del ambiente

Para que el proceso de parametrización esté completamente en control y no se den incidentes de distinta índole, se propone de acuerdo a la experiencia en ingeniería de software, que una persona sea la encargada de reunir toda la parametría por proyecto para que no se sobrescriban las mismas entre los distintos módulos. De igual manera esta persona ayudará a que cierta parametría no sea repetitiva y nos informe cuando una parametría ya existe en el ambiente. Este encargado debería ser nombrado en el Plan de Parametría del proceso conjuntamente con la definición sobre que ambientes se probará la parametrización desde cero.

Por otro lado, a cada miembro nuevo que se integre al equipo ágil del módulo, se le debería capacitar sobre cuál es la parametría básica que se tiene, y de esta manera evitar errores recurrentes al entregar la misma al encargado.

Finalmente los errores de parametría deberían ser cuantificados por el SCRUM Master en cuánto al tiempo de solución y a cuántas personas afectaron, de esta manera se tendría claro cómo afectaron al proyecto en curso.

4.1.6 Herramientas que mejoren eficiencia del desarrollador

Cuando se está desarrollando un software la rapidez con la que resuelvas problemas, errores y retos se traducen en dinero. Para corroborar lo antes dicho, el equipo ágil tomo el tiempo que se demora en encontrar un error, con y sin una herramienta de eficiencia y se redujo en un 50% en promedio el tiempo usado. Las tareas a resolver eran comparables en complejidad y el resultado siempre fue el mismo, el ahorro de un 50%. Si utilizaríamos dichas herramientas a lo largo de todas las capas de desarrollo de software (capa de base de datos, capa transaccional, capa de vista) podríamos realizar el doble de tareas aproximadamente usando el mismo tiempo. Además, teniendo en cuenta el uso de herramientas que nos ayuden a automatizar tareas como las pruebas, se ahorrarían recursos tanto de probadores como desarrolladores.

Tabla 9:
Tiempos promedios obtenidos al usar herramientas que incrementan la eficiencia

Tarea	Tiempo usando Herramientas	Tiempo sin usar Herramientas
• Buscar Errores en Base de Datos	0.5h	1h
• Comparar versiones para incluir funcionalidad	2h	4h
• Seguimiento a un error Transaccional	1.5h	3h

Continúa

Tarea	Tiempo usando Herramientas	Tiempo sin usar Herramientas
• Generar funcionalidad completa	8 h	16h
• Pruebas de la capa de Vista	0.5h	1h
• Pruebas de Carga transaccional	4h	8h
• Consulta y solución de error en base de conocimiento	0.5h	1h
• Generación de Diccionario de Datos para Documentación	0.5h	1h
• Pruebas de notificaciones por e-mail	1h	2h
• Preparación de Ambiente de Desarrollo	12h	24h

Tabla 10:
Tipos de Herramientas que incrementan Eficiencia

Tipos	Ejemplos
• Clientes y depuradores de Bases de Datos	Embarcadero, Aqua Data Studio
• Comparadores de Texto y Carpetas	Notepad ++, Araxis Merge
• Pruebas Automatizadas	Selenium, SoapUI, WatiN, JUnit
• Apertura de archivos grandes de texto optimizando recursos	Baretail
• Buscadores Avanzados de Texto	FileSeek, Baregrep
• Pruebas de Carga	FunkLoad, LoadUI, Jmeter
• Clientes FTP	Filezilla, Smart Ftp
• Generadores de Código	Genexus, Acceleo

Continúa

Tipos	Ejemplos
• Creación y configuración de Ambientes de Desarrollo propios	Vagrant, PuPHPet, Docker
• Soporte Remoto	TemViewer, Join me
• Documentación Automatizada	Javadoc
• Bases de conocimiento	Piggy, OpenKm, Twiki
• Pruebas de notificaciones en diferentes clientes de e-mail	Litmus
• Herramientas Intellisense, y de Productividad	Resharper, Eclipse
• Software para alimentación de Metodología SCRUM	Jira, Asana, iceScrum, Trello
• Modelamiento de Base de Datos e ingeniería inversa	Power Designer, Sql Power Architect, Druid, Open System Architect

Con el afán de lograr estas mejoras en cuanto a la eficiencia, la empresa ha empezado a definir dichas herramientas a nivel corporativo mediante una evaluación de las mismas de costo-beneficio. Dicha evaluación costo-beneficio se están discutiendo con los líderes de equipo que tienen muy claro los problemas que se viven a diario y las metas que exige la empresa año tras año. Esta mejora formaría una especie de punta de lanza en el giro cultural y de eficiencia que busca la empresa.

Para complementar dicha solución yo propongo que se maneje un proceso interno de capacitación para nuevos miembros en cada equipo ágil, donde se compartan claramente los estándares que mantiene la empresa, conocimientos

tecnológicos, funcionales y el uso de las herramientas e instalación de las mismas. Este proceso interno se lo podría aplicar para todas las áreas y módulos de la empresa por su generalidad y podría hacerse de manera transversal como los procesos de apoyo del proceso de desarrollo de software.

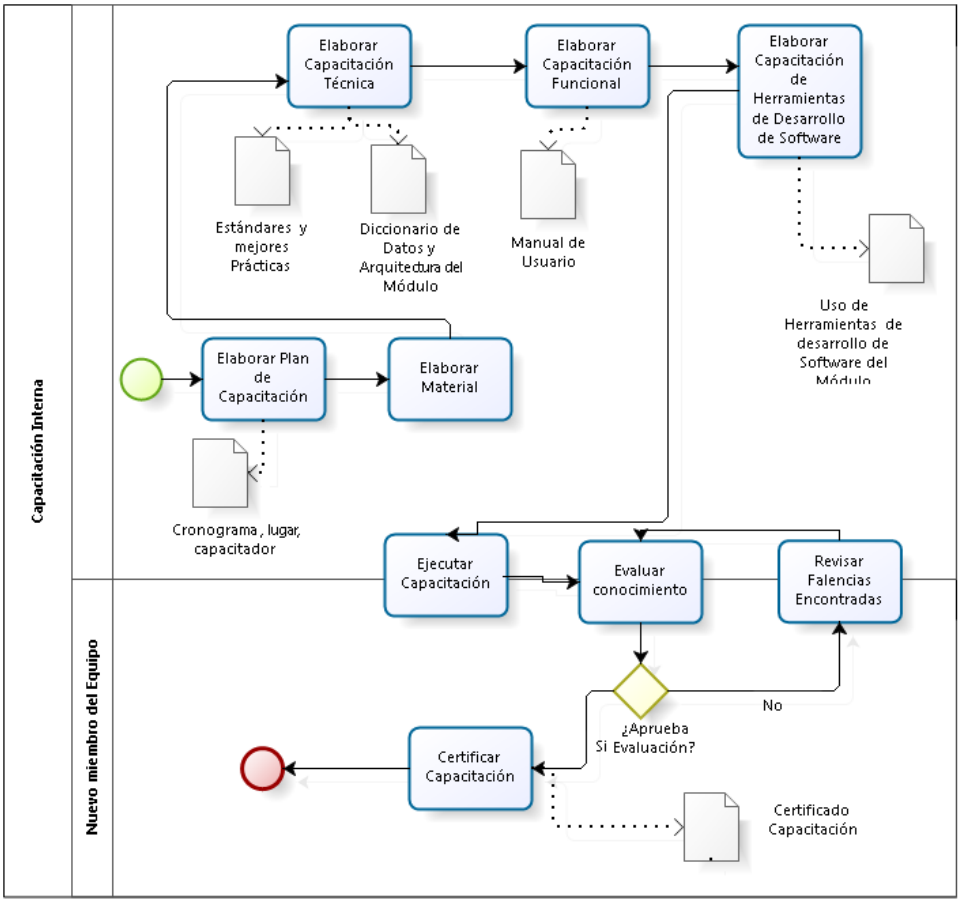

Figura 28 Proceso de Capacitación Interna dentro de célula ágil
Fuente. Daniel Jarrín, 2017

4.1.7 Indicadores del proceso

A continuación, detallo los índices que actualmente se tiene en la empresa conjuntamente con los de la propuesta:

Tabla 11:
Indicadores del proceso

Tipo	Categoría	Nombre	Fórmula
Indicadores Actuales	Metodología SCRUM	Velocidad del Equipo	Número de puntos de historia finalizados en un Sprint
		% Cumplimiento por Sprint	(Número puntos de historia aceptados /Número de puntos de historia comprometidos)* 100
		Valor del Negocio Entregado al Cliente	Valor total de negocio asociado con las funcionalidades entregadas en un Sprint
		% Deuda Técnica	(Valor de deuda técnica de Sprint expresada en puntos / Valor de puntos de historia realizados en Sprint)*100
	Productividad	Costo unitario Producción	Costo de célula por Sprint / Suma puntos de Historia realizados en Sprint
		Productividad por Sprint	((Número de funcionalidades expresadas en Puntos de Historia entregados por sprint * 8)/ Número de horas trabajadas equipo)
	Efectividad	Porcentaje de Cumplimiento de Trabajo objetivo Sprint	(Número de horas ejecutadas Sprint/ Número de horas objetivo planificadas Sprint)*100
	Eficiencia	Eficiencia Por Sprint	((Número de funcionalidades de sprint expresadas en Puntos de Historia planificados * 8)/ Número de horas trabajadas equipo)
		Costo por Horas Hombre en Sprint	Costo Total Célula en Sprint / Total Horas Ejecutadas
	Errores	Errores por sprint	Número de Errores encontrados en el sprint
Indicadores que se propone añadir	Productividad	Porcentaje de tiempo no trabajado en la célula por sprint	(Número de Horas no trabajadas en sprint / Número de horas planificadas objetivo en sprint)*100
		El tiempo de flujo (Time to Market)	Tiempo promedio que tarda una tarea desde que se asigna al equipo hasta que se termina
	Errores	Porcentaje de Errores por Deuda Técnica	(Número de errores causados por deuda Técnica en sprint / Número de errores totales en Sprint)*100
		Porcentaje de Tiempo usado en sprint por errores de deuda Técnica	(Tiempo en horas usadas en el Sprint para resolver errores de Deuda Técnica / Tiempo Total realizado en sprint por célula)*100
		Porcentaje de tiempo usado en sprint para resolver errores de control de fuentes	(Tiempo en horas usadas en el Sprint para resolver errores de control de fuentes / Tiempo Total realizado en sprint por célula)*100
		Porcentaje de tiempo usado en sprint para resolver errores de Parametrización	(Tiempo en horas usadas en el Sprint para resolver errores de Parametrización / Tiempo Total realizado en sprint por célula)*100
		Porcentaje de tiempo usado en sprint por errores de ambiente	(Tiempo en horas usadas en el Sprint causada por errores en ambientes / Tiempo Total realizado en sprint por célula) * 100
	Re-trabajo	Porcentaje de Cumplimiento de Atención por Error	(Número de Errores atendidos dentro tiempo / Número Total de Incidentes) * 100
		Porcentaje Devoluciones vs. Entregas de Errores	(Número de Devoluciones / Número de Errores Entregados) * 100

Se propone la creación de los 5 indicadores de errores, porque de acuerdo a la investigación se están consumiendo tiempos considerables para poder solucionar errores como deuda técnica, control de fuentes, parametrización y errores de creación y mantenimiento de ambientes. Cabe indicar que el rol encargado en recolectar los errores por tipo y el tiempo empleado en resolver los mismo será el líder técnico. Además, se propone la creación de dos indicadores más de productividad como son: porcentaje de tiempo no trabajado por célula al darse un error que afecta al equipo o a un miembro del equipo, y a tomar en cuenta el tiempo que demora la realización de una funcionalidad desde que se abre como requerimiento. Este indicador antes nombrado nos permitirá ganar una mayor cantidad de clientes en el futuro, conforme lo vayamos mejorando. La persona que se encargue de registrar el tiempo no trabajado del equipo en el sprint, causado por imprevistos es el líder técnico. Por último se proponen 2 indicadores más que permitirán controlar el re-trabajo y la disminución de los errores en el cliente como es uno de los objetivos en el área. Existe una herramienta que usamos para administrar nuestros proyectos, que nos permitirá calcular nuestros indicadores de re-trabajo

4.1.8 Esquema de mejora continua en el proceso

El esquema de mejora continua que se propone, se basa en la reunión de retrospectiva de la metodología SCRUM. A continuación, detallo los pasos a seguir:

1. Al realizar la reunión de retrospectiva se debe identificar el impedimento o problema más importante del Sprint para el equipo ágil. Por lo general, el impedimento más importante, podrá identificarse con los indicadores propuestos anteriormente. Se realizará un análisis del problema raíz y se propondrá soluciones.

2. Se debe añadir la solución del impedimento en la pila de tareas del siguiente Sprint para eliminar el mismo.

3. Este impedimento se lo traducirá a una historia de usuario que contendrá criterios de aceptación y se le planificara tiempos de solución. Es importante resaltar que si la historia de usuario es muy grande, se la debe dividir en tareas pequeñas que permitan el manejo adecuado en los siguientes sprints.

4. En la siguiente reunión retrospectiva, se deben evaluar los resultados de remover el impedimento, y seguir buscando un nuevo impedimento que nos permita mejorar de manera continua.

4.1.9 Matriz de impacto y facilidad de implementación de la mejora

Hemos priorizado el orden de implementación de las mejoras al proceso mediante una matriz de impacto y facilidad de implementación que se presenta a continuación:

IMPACTO

IMPLEMENTACIÓN

	ALTO	BAJO
FÁCIL	**1**	**3**
DIFÍCIL	**2**	**4**

Figura 29 Matriz de Impacto y Facilidad de Implementación
Fuente. Empresa Desarrolladora de Software analizada, 2016

Esta matriz tiene cuatro calificaciones que nos ayudarán a priorizar el orden de implementación de las mejoras a realizar:

1. Esta calificación se la da a la mejora que es de fácil implementación y tiene un alto impacto en el resultado del proceso.

2. Esta calificación se la da a la mejora que es de difícil implementación y tiene un alto impacto en el resultado del proceso.

3. Esta calificación se la da a la mejora que es de fácil implementación pero no tiene un alto impacto en el resultado del proceso.

4. Esta calificación se la da a la mejora que es de difícil implementación y no tiene un alto impacto en el resultado del proceso.

Siguiendo con la priorización, hemos asignado los valores de la calificación de dicha matriz a cada una de las mejoras propuestas, tomando en cuenta el tiempo que nos ahorramos si llegásemos a implementarla.

Tabla 12:
Calificación de Propuestas en base a Matriz de Impacto y Facilidad de Implementación

Propuesta	Calificación
• Mejora en la planificación de tiempos historias y tareas	1
• Mejora en la deuda técnica del equipo ágil	2
• Mejoramiento en el control del producto y subida de fuentes	1
• Correcta preparación de ambientes y disminución de daños	1
• Mejoramiento en la parametrización del ambiente	3
• Herramientas que mejoren eficiencia del desarrollador	1

4.2 VENTAJAS DE LAS MEJORAS PROPUESTAS

Las principales ventajas de la propuesta dada serían:

1. **Mejorar cumplimiento de tareas planificadas en Sprint:** Mediante las mejores prácticas de SCRUM en la planificación, y tomando en cuenta que el desarrollador debe conocer exactamente el estado actual de lo que va a modificar o a desarrollar, se busca minimizar la incertidumbre y mejorar el porcentaje de cumplimiento a 100% comparado con el 74% que se tiene en actualmente.

2. **Eliminar la Deuda Técnica y el tiempo invertido en la solución de este tipo de errores:** Mediante la asignación de un tiempo al líder técnico del equipo ágil para la revisión de errores técnicos, se busca poder reducir en un 20% los errores totales de un proyecto, y por otro lado automatizar todas las pruebas en las distintas capas del software. Esta disminución conjuntamente con las pruebas automatizadas, provocarían una disminución del 3% al tiempo total invertido en un proyecto.

3. **Eliminar errores de versión de producto no controlada:** Mediante la capacitación a los nuevos miembros del equipo sobre la importancia de subir correctamente los fuentes del software, conjuntamente con el compromiso del equipo a resolver los errores de compilaciones automáticas, nos aseguramos el disminuir los errores causados por una versión de proyecto no controlada, y confiamos que al descargar los fuentes para su modificación, tendremos la última versión correcta. Estas acciones correctivas nos permitirán disminuir en un 3% aproximadamente, el tiempo empleado en el desarrollo de un proyecto de software para el módulo de Banca en Línea.

4. **Reducir errores de preparación de ambientes con sus correspondientes retrasos:** Mediante la creación de las solicitudes del ambiente por parte de un especialista del módulo conjuntamente con el Scrum Master, se logrará disminuir errores de faltantes y de desconocimientos de la versión de ambiente que se quiere obtener. Además, mediante la asignación de accesos para cada uno de los usuarios del ambiente, nos aseguraremos que se disminuyan los errores en los mismos por una incorrecta utilización. En la

actualidad, estos errores ocasionan grandes retrasos no solo a un equipo sino a varios por ser ambientes compartidos. Para este estudio se llegó a comprobar que se podría disminuir en un 3% el tiempo total empleado de un proyecto de software.

5. **Reducir errores de parametrización con sus correspondientes retrasos:** Mediante la centralización de la parametrización sobre una persona, que básicamente se encargue de tener listos los artefactos de instalación, y de informar la parametrización lista a los distintos módulos, lograremos disminuir errores de borrado de parametrización conjuntamente con la sobre escritura de la misma. También, nos enfocaremos en la capacitación de la parametrización básica por cada módulo, para poder iniciar un desarrollo o al incluir una nueva persona al equipo ágil. Actualmente la solución de este tipo de errores representan un 2% del tiempo total empleado en el desarrollo del proyecto.

6. **Incrementaremos la eficiencia y productividad del equipo ágil:** Mediante la definición de herramientas que faciliten el trabajo al desarrollador y sean utilizadas en toda la empresa y por cada módulo específico, se espera reducir en un 50% el tiempo utilizado para desarrollar o resolver un error, de acuerdo a la experiencia desarrollada. Esta mejora tendrá efecto, siempre y cuando a las personas nuevas se les siga transmitiendo el conocimiento de dichas herramientas con capacitaciones. Además, se ha definido la creación de un documento que trate sobre el uso de dichas herramientas a nivel del módulo de Banca en Línea.

7. **Uso de nuevos indicadores que ayuden al control del proceso y complementen nuestra propuesta:** Mediante la definición de nuevos indicadores para el proceso de desarrollo de software del módulo de Banca en Línea, se busca tomar en cuenta temas como re-trabajo, falta de productividad, y errores que en la actualidad están quitando tiempo de desarrollo al equipo y no permiten tener un proceso más controlado.

8. **Propuesta promueve la mejora continua:** Basándonos en el uso de las reuniones de retrospectiva propias de la metodología SCRUM, se ha propuesto un esquema que permite al módulo de Banca en Línea seguir mejorando iteración tras iteración al analizar y resolver los principales impedimentos o problemas con el equipo ágil. Normalmente, la metodología SCRUM ya propone las reuniones de retrospectiva como componente para remover los impedimentos y problemas que se dieron en la iteración, sin embargo, la propuesta busca hacer uso de los indicadores propuestos, con el fin de tener una base cuantitativa para realizar las mejoras. Los integrantes del equipo ágil debemos comprometernos a resolver los problemas que se identifiquen para conseguir la mejora continua que se espera.

9. **Propuesta afecta directamente la productividad mediante la reducción del tiempo:** La fórmula general de la productividad de un proceso es:

   Productividad = Bienes o servicios producidos / Recursos usados

Debido a que una empresa de software no produce algo tangible, sino más bien produce puntos de historia según SCRUM, que reflejan características o funcionalidades de software, la empresa estudiada usa el indicador:

Productividad = ((Número de funcionalidades expresadas en Puntos de Historia entregados por sprint * 8) / Número de horas trabajadas equipo)

Tomando en cuenta los tiempos reducidos en la propuesta por deuda técnica en 3%, versión de producto no controlada en 3%, preparación de ambientes en 3%, parametrización en 2%, y un 50% por uso de herramientas de productividad y eficiencia, sumarían un total de 61% reducido en tiempo. Esta reducción de tiempo o horas trabajadas del equipo, afectarían directamente a la productividad del mismo durante los sprints.

4.3 DESVENTAJAS DE LAS MEJORAS PROPUESTAS

Las principales desventajas de la propuesta dada serían:

1. **Apoyo de la alta gerencia para la implementación de la mejora:** Como cualquier cambio a nivel de empresa, se requiere que la alta gerencia se comprometa a dar el total apoyo hacia la implementación de las mejoras y la continua vigilancia para que se cumplan los pasos dados conjuntamente con la correcta actualización de los indicadores, los cuales nos servirán para controlar el proceso de mejor manera.

2. **Muchos de los cambios de la propuesta dependen de una correcta capacitación:** Varios de los cambios de la propuesta serán efectivos siempre y cuando las capacitaciones sean preparadas, actualizadas y entregadas correctamente por el especialista técnico de la célula ágil. Por otro lado, la recepción y puesta en práctica dependerá de la actitud y apertura del alumno para identificar la importancia y uso de la misma a lo largo del desarrollo de sus tareas diarias.

3. **Cambios de la mejora propuesta involucran más trabajo y compromiso de equipo:** Como en cualquier cambio empresarial, se requiere que la célula se comprometa a cumplir todo el trabajo de actualizar correctamente los indicadores, identificar los principales problemas y resolverlos. Es decir, se trata de dar un giro de cultura empresarial hacia la mejora continua que muchas de las veces cuestan tiempo y esfuerzo extra a cada equipo ágil de la empresa en estudio.

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La propuesta de mejora del proceso de desarrollo de software se basó en la metodología de los 7 pasos o más conocido como MP. Los pasos son: definir los límites del proceso, observar los pasos del proceso, recolectar los datos relativos al proceso, analizar los datos recolectados, identificar las áreas de mejora, desarrollar mejoras, implantar y vigilar las mejoras. Al momento de recolectar los datos relativos al proyecto, se usó la metodología del caso único, debido a que no se tenían más datos históricos de proyectos dentro de la empresa, puesto que la duración de los mismos es generalmente semestres o años. La metodología de caso único tiene las ventajas de proveer la capacidad de establecer las condiciones y naturaleza de una relación entre causa-efecto, la capacidad de trabajar en situaciones atípicas por motivos como los antes descritos, pero con las desventajas de no poder generalizar sus propuestas o conclusiones, debido a que estadísticamente un muestreo mayor provee mayor confiabilidad, y la variabilidad por diversos factores de los proyectos dentro de la empresa de software pueden ocasionar cierto error en las inferencias del estudio. Por otro lado, con las limitantes sobre las tareas dentro de los procesos de desarrollo de software que no son comparables entre sí, ni tienen tiempos estándar establecidos, se tuvieron que usar herramientas de calidad como diagrama de causa y efecto, lluvia de ideas para identificar los puntos a mejorar en el proceso de desarrollo de software del módulo de Banca en Línea.

- La realidad del proceso de desarrollo de software en el módulo de Banca en Línea es que se han encontrado falencias principalmente en el seguimiento de errores que se producen dentro del mismo. Los principales errores identificados en el proceso son: deuda técnica alta, incorrecto control de fuentes, errores de parametría, errores de preparación y funcionamiento de ambientes. Con el fin de tener un mejor control sobre los errores antes nombrados, se han definido 5 nuevos indicadores que nos permiten calcular el tiempo total empleado por sprint en resolver cada tipo de error. Además, estos errores serán tomados en cuenta en las reuniones de retrospectiva, para buscar una mejora continua con respecto a los mismos, o tomar decisiones estratégicas para su disminución. Por otro lado, se han definido 4 nuevos indicadores que complementará la medición de temas de retrabajo y productividad para la empresa.

- La revisión del código fuente por parte del líder técnico del equipo, conjuntamente con la implementación de las pruebas automatizadas a la capa de vista y de código fuente dadas en la propuesta, nos permitirán eliminar la deuda técnica del 68% y a su vez reducir en 3% el tiempo empleado en un proyecto. Además, nos permitirá erradicar aproximadamente el 20% de los errores totales cometidos en el desarrollo del proyecto.

- Al asignar la creación de las solicitudes de un nuevo ambiente al especialista del módulo, el cual tiene una visión clara sobre el detalle de los requerimientos del mismo, y por otra parte la asignación de accesos a servidores y bases de datos a cada miembro del equipo ágil, nos permitirá reducir en un 3% aproximadamente el tiempo total empleado en el desarrollo de un proyecto de software. Esta

disminución se dará al eliminar los errores de preparación de ambientes y daños a los mismos por el trabajo diario.

- Las compilaciones automáticas del código fuente, apoyadas en las capacitaciones iniciales a nuevos miembros del equipo, sobre la importancia de la correcta subida del código fuente permitirán reducir en un 3% el tiempo en que empezará el desarrollo de un proyecto. Cabe recalcar que el indicador de time to market será directamente afectado debido a que el desarrollo de una nueva funcionalidad empezará más rápido.

- Al centralizar la parametrización de los ambientes en una persona única, que se encargue de no repetir y sobrescribir parametría entre módulos, se disminuirá en un 2% aproximadamente el tiempo total del desarrollo de software de un proyecto. Además, esta propuesta debe ir de la mano con la capacitación a los nuevos miembros del equipo, sobre la parametría básica que se debe entregar a la persona que centraliza la misma.

- Se busca cumplir en un 100% con las tareas planificadas de cada entrega parcial o sprint, al permitir analizar al equipo de desarrolladores el estado actual de las tareas que se quieren realizar. Además, mediante el uso de mejores prácticas de SCRUM, se quiere cambiar el día de planificación a los días martes, debido a que existe menos ausentismo y el equipo se encuentra más atento que los días lunes de inicio de sprint. Actualmente, se tiene un promedio de cumplimiento del 74%.

- El uso de herramientas que incrementen la eficiencia al momento de desarrollar software a nivel corporativo, nos permitirán disminuir aproximadamente en un 50% el tiempo total empleado para el desarrollo de software. En la propuesta se detalla una lista de herramientas que abarcan temas de comparadores de texto, depuradores de bases de datos, generadores de código fuente, pruebas automatizadas, creadores de ambientes locales, etc.

- La mayoría de las soluciones y mejoras al proceso de desarrollo de software del módulo de Banca en línea, nacen de los mismos desarrolladores que viven día a día los problemas que se generan en su trabajo. En este sentido, se debe buscar que las reuniones de retrospectiva de la metodología SCRUM, sea un espacio donde todos los equipos ágiles analicen a conciencia los problemas e impedimentos tenidos en el sprint y busquen soluciones viables en corto tiempo. En la propuesta se detalla un esquema de mejora continua, basado en las reuniones de retrospectiva, que permite al módulo de Banca en Línea seguir mejorando sprint tras sprint al analizar y resolver los principales impedimentos o problemas con el equipo ágil.

- La propuesta afecta directamente a la productividad mediante la reducción del tiempo que se usa en el desarrollo de software, debido a que el tiempo que lograríamos ahorrar por la propuesta ascendería a un 61%. Actualmente la empresa cuenta con un indicador de productividad y este será realmente incrementado con el ahorro de tiempo de la propuesta.

5.2 RECOMENDACIONES

- En caso de que la empresa quiera ejecutar la mejora dada en el presente trabajo de titulación, se recomienda que fundamente la ejecución de la misma, en la matriz de facilidad de implementación e impacto, de esta manera se podrán ver resultados de forma rápida y ágil.

- Se debe tomar en cuenta que las conclusiones a las que se llegó en este trabajo de titulación no pueden ser generalizadas a todos las áreas, módulos y submódulos de la empresa, sin el estudio de una muestra más grande y significativa, puesto que por la falta de datos, se tuvo que realizar el estudio basado en la metodología del caso único.

- En caso de querer implementar cualquier cambio en un proceso organizacional, se debe contar con el total apoyo de la gerencia, de esta manera nos aseguramos que el cambio impacte a toda la empresa y todos vayamos alineados con los objetivos estratégicos de la misma.

- Se recomienda tener en cuenta las mejores prácticas y cambios a nivel de la metodología SCRUM, para de esta manera aprovechar al máximo esta tendencia en la manera de trabajar con proyectos de Software.

- Recomiendo se busque realizar las reuniones de retrospectiva a nivel de todas los equipos y módulos de la empresa, puesto que estas reuniones han servido

para generar varias de las soluciones de la propuesta tratada en este trabajo de titulación.

- Recomiendo se dé gran importancia a las capacitaciones técnicas y funcionales dentro de los equipos ágiles, puesto que las metodología SCRUM y el marco de referencia Scaled Agile Framework, basan su efectividad en el conocimiento de las tareas que se van a realizar.

- Se recomienda tener en cuenta las mejores prácticas y cambios a nivel del marco de referencia Scaled Agile Framework, para aprovechar al máximo las ventajas que pueda entregarnos.

- Se recomienda la implementación de los KPA’s del nivel administrado o 4 para poder seguir mejorando la madurez del proceso con respecto al modelo CMM.

BIBLIOGRAFÍA

Abad, J. (12 de Julio de 2016). *www.lecciones-aprendidas.info*. Obtenido de http://www.lecciones-aprendidas.info/search?q=Hablemos+de+agilidad

Acuña, K. (2009 de Agosto de 2009). Obtenido de http://www.eumed.net/: http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

AESOFT. (28 de Julio de 2004). *Biblioteca Banco Central del Ecuador*. Recuperado el 7 de Enero de 2017, de http://biblioteca.bce.ec/cgi-bin/koha/opac-detail.pl?biblionumber=56425

AESOFT. (13 de Febrero de 2015). *AESOFT (Asociación Ecuatoriana de Software).* Obtenido de AESOFT: https://issuu.com/aesoftecuador/docs/catalogo_aesoft_2015.pdf

AESOFT. (7 de Enero de 2016). *AESOFT*. Recuperado el 17 de Enero de 2015, de http://aesoft.com.ec/?page_id=38

Barker, I. (14 de Marzo de 2016). *BetaNews*. Obtenido de http://betanews.com/2016/03/30/firewalls-and-network-security/

Barret, S. (9 de Abril de 2014). *STEVE BARRETT'S BLOG*. Obtenido de https://barrettsteve.wordpress.com/2014/04/09/scaling-agile-using-the-scaled-agile-framework-safe/

Blog la tecla de Escape - Ingeniería de Software. (29 de Julio de 2015). *Blog la tecla de Escape*. Obtenido de http://latecladeescape.com/h/2015/07/metodologias-de-desarrollo-del-software

Briceño, E. (23 de Octubre de 2012). *Hipertextual*. Obtenido de https://hipertextual.com/archivo/2012/10/consejos-seguridad-para-empresas/

Carriedo, C. (12 de Agosto de 2015). *Forbes México*. Obtenido de http://www.forbes.com.mx/4-tips-para-llevar-tu-empresa-a-procesos-eficientes/#gs.dt8KLTQ

El Comercio. (7 de Noviembre de 2014). *El Comercio*. Recuperado el 7 de Enero de 2016, de http://www.elcomercio.com/actualidad/software-ecuador-encuentro-ministerio-telecomunicaciones.html

Garzás, J. (25 de Septiembre de 2013). Obtenido de http://www.javiergarzas.com/2013/09/scaled-agil.html

Garzás, J. (2 de Octubre de 2013). Obtenido de http://www.javiergarzas.com/2013/10/wsjf-safe.html

Gilibets, L. (31 de Julio de 2013). *Comunidad IEBS*. Obtenido de http://comunidad.iebschool.com/iebs/general/metodologia-kanban/

Gómez, N. (27 de Junio de 2009). *Calidad y gestión empresarial. ISO 9001 e ISO 14001*. Obtenido de http://hederaconsultores.blogspot.com/2009/06/enfoque-procesos-principios-iso-9001.html

González, H. (11 de Julio de 2012). *Calidad y Gestión*. Obtenido de https://calidadgestion.wordpress.com/2012/07/11/herramientas-para-la-mejora-continua/

Gutiérrez, C. (20 de Enero de 2014). Obtenido de http://www.i2btech.com: http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metogologia-agil/

Hundermark, P. (2011). Reuniones de Sprint. En S. c. Hundermark, *Un Mejor SCRUM* (pág. 9). Buenos Aires: ScrumSense.

ISO 9001. (2008). *ISO 9001 Norma Internacional.* Ginebra: Secretaría Central de ISO.

ItPuebla. (30 de Enero de 2011). *ItPuebla*. Obtenido de https://itpuebla.wordpress.com/2011/01/30/encriptacion/

Leffingwell's, D. (13 de Junio de 2016). *SAFe Scaled Agile Framework.* Obtenido de http://www.scaledagileframework.com/

Maldonado, A. (23 de Junio de 2013). *Gestión de Procesos*. Recuperado el 25 de Octubre de 2015, de Academia: https://www.academia.edu/10342201/GESTI%C3%93N_DE_PROCESOS

Neothek. (Junio de 6 de 2009). *Blog Neothek.* Obtenido de http://blog.neothek.com/blog-neothek/que-es-un-certificado-ssl-y-como-funciona/

Normas ISO 9001. (15 de Noviembre de 2008). *normas9000.com*. Recuperado el 25 de Octubre de 2015, de http://www.normas9000.com/iso-9000-59.html

PMOInformatica.com. (29 de Agosto de 2012). *PMO Informática Oficina de proyectos*. Obtenido de http://www.pmoinformatica.com/2012/08/5-metricas-para-proyectos-de-desarrollo.html

ProyectosAgiles.org. (10 de Octubre de 2016). *ProyectosAgiles.org*. Obtenido de https://proyectosagiles.org/historia-de-scrum/

Rayo, Á. (27 de Junio de 2017). *Lean Software Development (LSD): los siete principios*. Obtenido de http://www.netmind.es/knowledge-center/lean-software-development-lsd/

Romeu, A. (9 de Octubre de 2014). *Blog del Informático*. Obtenido de http://albertoromeu.com/scrum-planning-poker/

Schwaber, K., & Sutherland, J. (1 de Julio de 2013). *http://www.scrumguides.org/.* Obtenido de http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf

Suarez Rojas, D. A. (20 de Agosto de 2013). Obtenido de https://www.academia.edu: https://www.academia.edu/4324067/Metodolog%C3%ADas

Varguez Cahuich, M. A. (19 de Marzo de 2014). *Aplicación de estándares de calidad.* Obtenido de Aplicación de estándares de calidad: http://mitareaestandaresdecalidad.blogspot.com/2014_03_01_archive.html

Villaseñor, B. (6 de Marzo de 2013). *Uhma Blog*. Obtenido de https://www.uhmasalud.com/blog/bid/274000/Tips-para-tu-proceso-de-mejora-continua

ANEXOS

ANEXO 1. TIPS PARA IMPLEMENTAR PROCESOS EFICIENTES

De acuerdo a (Carriedo, 2015), la implementación de procesos que disminuyan errores y eliminen el tiempo que se usa en actividades que generan poco valor agregado se vuelven fundamentales a la hora de pensar en los tres pilares fundamentales de una empresa: accionistas, empleados y clientes. Regularmente la manera más fácil para saber cuándo documentar y estandarizar un proceso llega como retroalimentación de alguno de los tres pilares antes nombrados.

Según (Carriedo, 2015), los cambios para implementar procesos eficientes son:

- Definir prioridades tomando en cuenta los procesos críticos de la organización, escuchando a clientes, accionistas y empleados. Más tarde se deberán asignar responsables, fechas y tiempos de entrega.

- Definir formatos para todos los documentos que ayuden a describir desde lo general a lo particular. Estos formatos deben ser muy fáciles e intuitivos de llenar.

- Nombrar a un líder o responsable de área que documente a detalle lo sucedido en el funcionamiento del proceso, de esta manera se podrán detectar oportunidades de mejora.

- Una vez realizados los tres pasos anteriores se debe ejecutar lo documentado. En la ejecución todos los involucrados se guían con la metodología recopilada y prácticas más exitosas. Con base en éstas se puede dar inicio a la capacitación del personal para más tarde buscar ajustes e implementar mejoras.